TargetLink Data Dictionary

Basic Concepts Guide

For TargetLink Data Dictionary 5.1

Release 2020-B – November 2020

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 Guide

Contents

This guide gives you all the information you need on the TargetLink Data Dictionary, for example, its benefits and how it is used in the dSPACE tool chain. The guide also explains how to manage data objects with the TargetLink Data Dictionary Manager. In addition, you will learn how to manage data objects with the Data Dictionary MATLAB API.

Orientation and Overview Guide For an introduction to the use cases and the TargetLink features that are related to them, refer to the 📖 TargetLink Orientation and Overview Guide.

Required knowledge

This guide is most useful to function developers and software specialists working with TargetLink.

Knowledge in handling the host PC and the Microsoft Windows operating system is assumed.

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 Adobe® 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.

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.

Introduction to the TargetLink Data Dictionary

Introduction

The **TargetLink Data Dictionary** is a central data container that holds all the relevant information about an ECU application, for example, for code generation.

Where to go from here

Information in this section

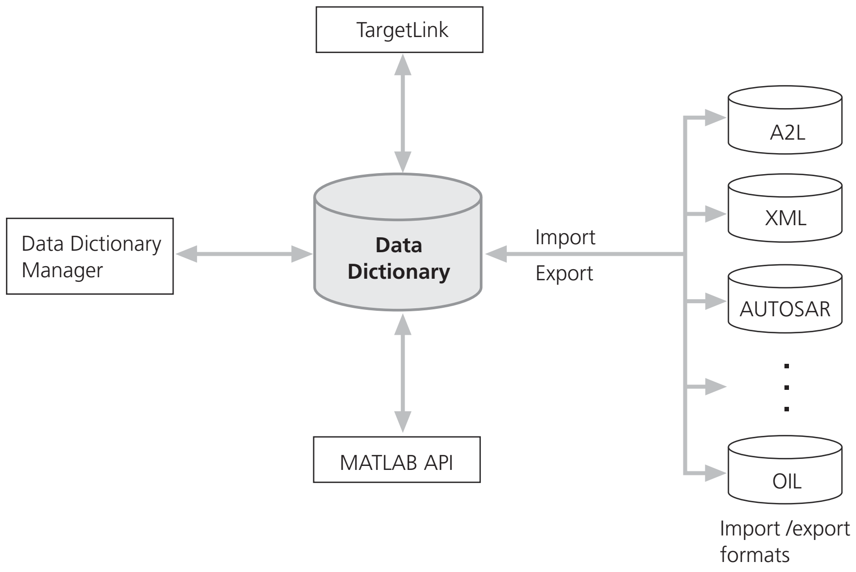
Basics on the TargetLink Data Dictionary

Introduction

The **TargetLink Data Dictionary** is a central data container that holds all the relevant information about an ECU application, for example, for code generation.

Data Dictionary objects can be referenced from Simulink models independently of any model partitioning to keep data consistent throughout all stages of the development process. You can define variables and properties. Structured data types can be specified and used for variable declarations. Scaling formulas can be entered and used to uniformly scale fixed-point signals and parameters in the model. The **TargetLink Data Dictionary** holds a wealth of additional information, for example, specifics on function calls, tasks, variable classes, data variants, generated C modules and so forth. The data is well structured in a tree and can be accessed via application programming interfaces (APIs).

The following illustration shows the **TargetLink Data Dictionary** with the interfaces for data management and data import/export.



When a data dictionary (DD) is loaded from file, its contents are loaded into the DD workspace, which is directly mapped into the memory space of the calling process. Thus access times to DD objects are performance-optimized. Application programs like TargetLink work with independent instances of the Data Dictionary. This means that changes made in another application are not visible in TargetLink until they are back-propagated to TargetLink as a DD project file or when a DD import module is used.

Keep in mind that the Data Dictionary is not a central database server with multi-user access. For information on how to work with the Data Dictionary, refer to Basics on the Data Dictionary Manager on page 31.

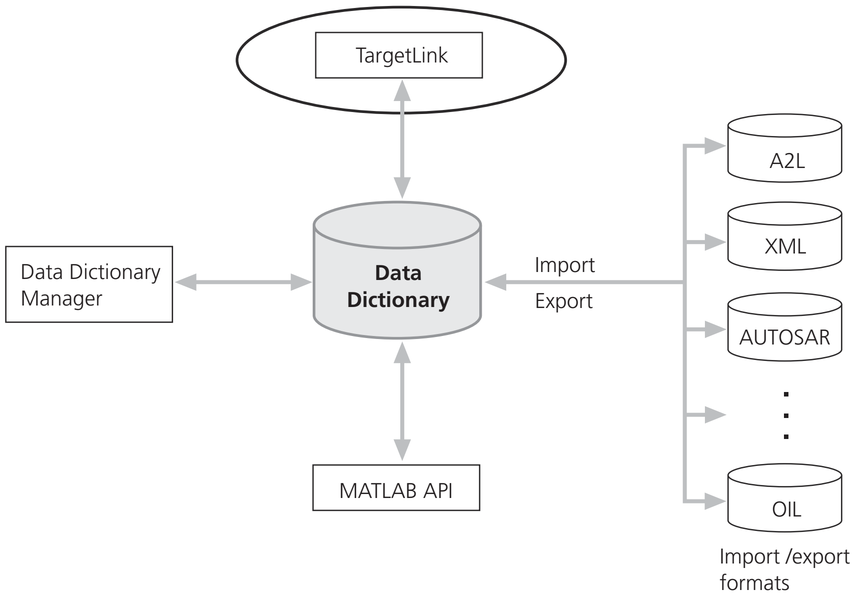
Related topics

Basics

Basics on References to Data Dictionary Objects in Block Dialogs

Introduction

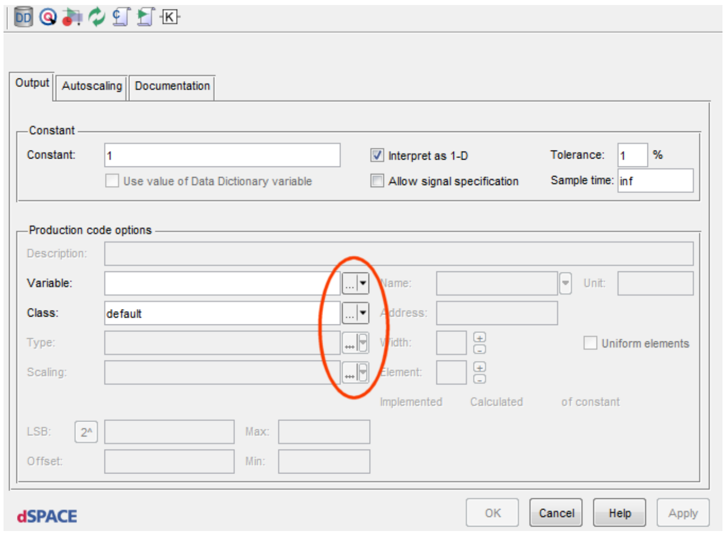
TargetLink uses the **TargetLink Data Dictionary**.



TargetLink

TargetLink blocks and Stateflow objects can reference DD objects in the **TargetLink Data Dictionary**. Working with DD objects gives you 3 main advantages:

- You have consistent data definitions not only in TargetLink models, but also in all stages of development.
- You can separate an algorithm defined in the Simulink model from the actual parameters stored in the **TargetLink Data Dictionary** (clear separation of data and algorithm).
- You can easily import your data into the **TargetLink Data Dictionary** and make it available in the dSPACE tool chain, for example, in TargetLink block dialogs, which provide access to these data via browse buttons.



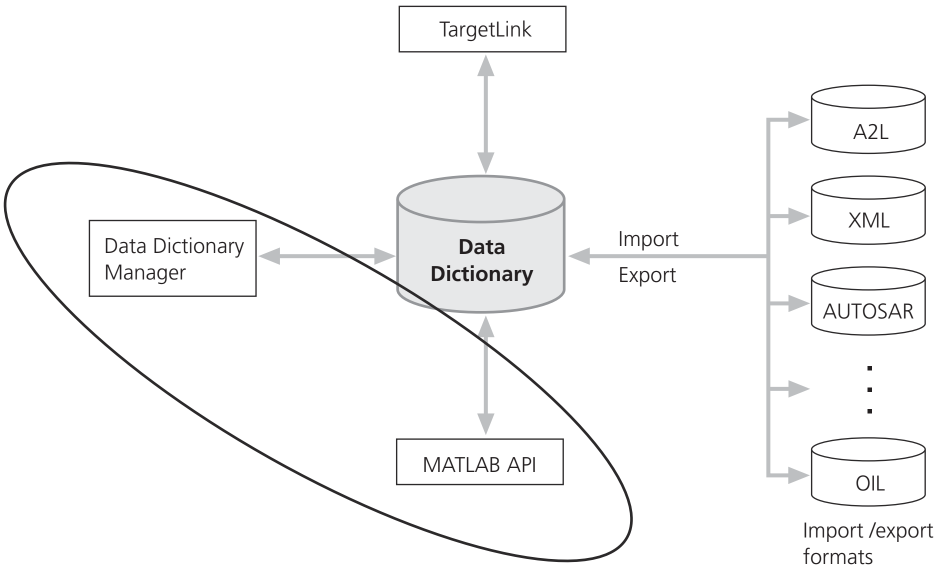
Constant: 1
[✓] Interpret as 1-D
Tolerance: 1 %
[ ] Use value of Data Dictionary variable
[ ] Allow signal specification
Sample time: inf
Variable:
Class: default
[ ] Uniform elements

After code generation, TargetLink writes a detailed description of the resulting C code to the **TargetLink Data Dictionary**, including lists of generated C modules, functions, variables, type definitions, scalings, variable classes, etc. This information can be used later to run postprocessing tools, or to perform automated unit tests.

Basics on Accessing the Data Dictionary via GUI or API

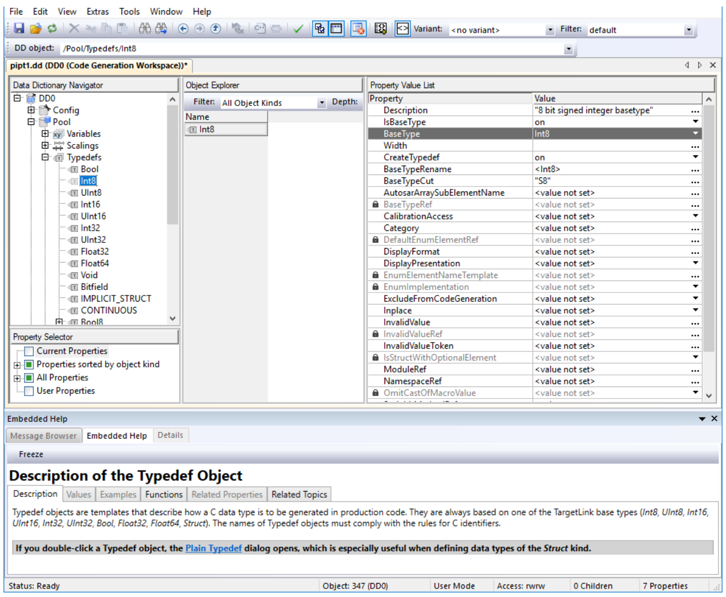
Introduction

Typical user interface functions are available, like creating, editing, copying and pasting objects, loading and saving individual data branches and searching the entire dictionary.



TargetLink Data Dictionary Manager

The TargetLink Data Dictionary Manager is the graphical user interface for the TargetLink Data Dictionary. It provides a browser to explore the DD object tree and dialogs to manipulate DD objects, or to call import/export functions.



Property	Value
Description	"8 bit signed integer basetype"
IsBaseType	on
BaseType	Int8
Width	
CreateTypedef	on
BaseTypeRename	<Int8>
BaseTypeCut	"S8"
AutosarArraySubElementName	<value not set>
BaseTypeRef	<value not set>
CalibrationAccess	<value not set>
Category	<value not set>
DefaultEnumElementRef	<value not set>
DisplayFormat	<value not set>
DisplayPresentation	<value not set>
EnumElementNameTemplate	<value not set>
EnumImplementation	<value not set>
ExcludeFromCodeGeneration	<value not set>
Inplace	<value not set>
InvalidValue	<value not set>
InvalidValueRef	<value not set>
InvalidValueToken	<value not set>
IsStructWithOptionalElement	<value not set>
ModuleRef	<value not set>
NamespaceRef	<value not set>
OmitCastOfMacroValue	<value not set>

Description of the Typedef Object

Description | Values | Examples | Functions | Related Properties | Related Topics

Typedef objects are templates that describe how a C data type is to be generated in production code. They are always based on one of the TargetLink base types (*Int8, UInt8, Int16, UInt16, Int32, UInt32, Bool, Float32, Float64, Struct*). The names of Typedef objects must comply with the rules for C identifiers.

If you double-click a Typedef object, the Plain Typedef dialog opens, which is especially useful when defining data types of the *Struct* kind.

Status: Ready | Object: 347 (DD0) | User Mode | Access: rwrw | 0 Children | 7 Properties

The TargetLink Data Dictionary is protected by a license which is bundled with TargetLink. You must purchase the license and activate it. Refer to License Activation (📖 Working with CodeMeter Licensing Technology).

MATLAB API

The TargetLink Data Dictionary MATLAB API (application programming interface) gives you access to the TargetLink Data Dictionary via MATLAB. All the functions necessary for managing data are available, for example, for creating DD objects and defining their properties. You can use these commands in the MATLAB Command Window or in MATLAB M files. For details, refer to Data Dictionary MATLAB API.

Example

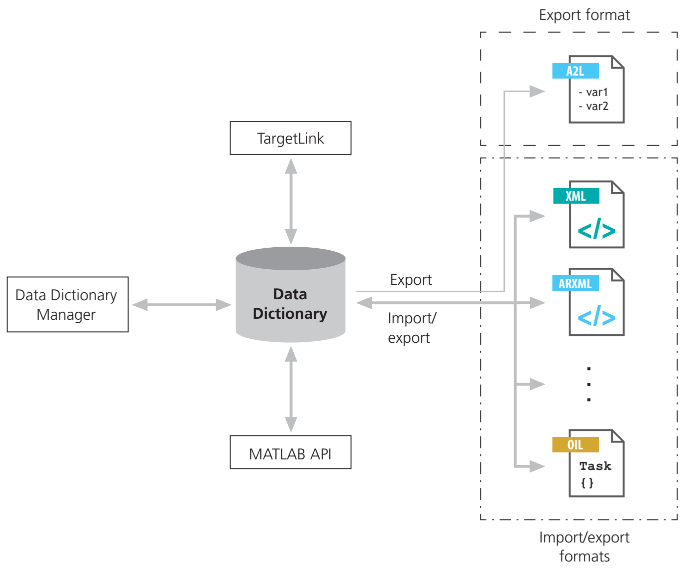
This example shows how the MATLAB API functions are used to create a new Variable object:

```
hSpeed = dsdd('AddVariableObject','/Pool/Variables','speed');
dsdd('Set',hSpeed, ...
    'Description','Engine speed', ...
    'Type', 'UInt16', ...
    'Class','DISP' ...
    );
```

Basics on Importing and Exporting Data

Introduction

You can import data into the TargetLink Data Dictionary and export data from it to files in different file formats.



File formats

In this version of the TargetLink Data Dictionary, the following file formats are supported:

- ASAM MCD-2 MC files (also known as ASAP2 files)
- XML files
- OIL files
- AUTOSAR

Own database

If you already manage your data in a database in your own file format, you can add your data to the TargetLink Data Dictionary. The MATLAB API provides commands to create and manage all types of DD objects. You can use the DD MATLAB API to develop your own export or import interfaces to interface your own database to the TargetLink Data Dictionary. You then have direct access to your data via TargetLink.

Related topics

Basics

Basics on Importing and Exporting Data Between MATLAB and DD Variable Objects (📖 TargetLink Interoperation and Exchange Guide)
Exporting Variable Objects (📖 TargetLink Interoperation and Exchange Guide)
Importing Variable Objects (📖 TargetLink Interoperation and Exchange Guide)

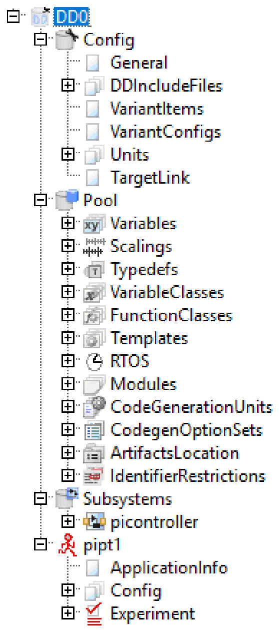
HowTos

How to Export Data Files from DD Workspaces.. 51
How to Import Data Files to DD Workspaces... 50

Basics on the Data Dictionary Structure (DD Object Tree)

Data storage according to purpose and origin

The TargetLink Data Dictionary has main areas for setting configuration data, pool data for models (pre-code generation data), subsystem data and application data (post-code generation data). The DD's structure and navigation are comparable to a registry, which means that property data and their values are placed in a tree (called the DD object tree), see the following illustration.



```
DD0
  Config
    General
    DDIncludeFiles
    VariantItems
    VariantConfigs
    Units
    TargetLink
  Pool
    Variables
    Scalings
    Typedefs
    VariableClasses
    FunctionClasses
    Templates
    RTOS
    Modules
    CodeGenerationUnits
    CodegenOptionSets
    ArtifactsLocation
    IdentifierRestrictions
  Subsystems
    picontroller
  pipt1
    ApplicationInfo
    Config
    Experiment
```

DD objects

The objects of this tree are the DD objects. Each object has one parent, except the DD root object, and a specified number of child DD objects. The Data Model specifies which DD objects can be parented by which DD objects. In addition to child objects, a DD object can have properties, whose names and possible values are also specified in the Data Model.

Main areas

The DD object tree is divided into five main areas:

Config The root object of the Config area is called Config. It is a child object of the DD root object. It contains configuration data for the tools working with the TargetLink Data Dictionary and configuration data for the TargetLink Data Dictionary itself. The configuration data for the TargetLink Data Dictionary contains a list of included DD files, data for variant configurations, etc. The data in the Config area is typically maintained by a Data Dictionary administrator.

Pool The root object of the Pool area is called Pool. It is a child object of the DD root object. The Pool area contains all input data required for code generation, for example, definitions of calibratable and measurable variables, type definitions, and RTOS objects. You can define the hierarchy of the Pool area to structure the data. All DD references from TargetLink blocks refer to DD objects in the Pool area. You will mainly work with Pool objects during the model design phase.

Subsystems The root object of the Subsystems area is called Subsystems. This object consists of an arbitrary number of Subsystem objects, each of which is the result of code generation for a specific Simulink subsystem. The Subsystem objects contain detailed information on the generated code, including C modules, functions, etc. The data in the Subsystems area is either automatically generated, or imported from ASAM MCD-2 MC. You must never modify DD objects in the Subsystems area because this causes inconsistencies with the generated code.

<Application> An optional DD object that is a child object of the DD root object. Each Application object defines how an ECU program is built from the generated subsystems. It also contains some experiment data, for example, a list of variables to be logged during simulations and results of code coverage tests. Build objects are children of Application objects. They contain all the information about the binary programs built for a certain target platform, for example, the symbol table for address determination.

<UserData> Optional UserData objects and user-defined properties can be attached to many DD objects, including the DD root object. This is a generic mechanism to add your own company-specific data.

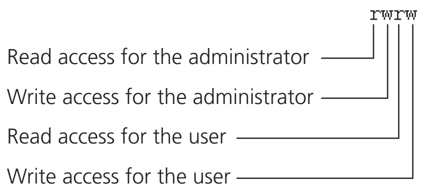
Basics on User Modes and Access Rights

2 user modes

The TargetLink Data Dictionary supports two different modes of access rights. These modes define which DD objects are readable or writable.

Access rights of DD objects

The DD objects are protected by access rights. The TargetLink Data Dictionary supports two modes: an administrator mode (admin access mode) and a user mode (user access mode). The administrator mode can be protected by a password which is stored with the DD project file. In the Data Dictionary the access rights are displayed in the *rwrw* format, which has the following meanings:


rwrw
Read access for the administrator
Write access for the administrator
Read access for the user
Write access for the user

The access rights of the DD objects can be defined individually for each item in the DD object tree. There is no inheritance mechanism which propagates access rights from parent to child objects. The access rights are independent of the DD object tree. This means that a parent object can have access rights other than its child objects. However, you can use the Data Dictionary Manager or DD MATLAB API to set the access rights of a DD object or an entire DD object subtree.

The DD objects in the Subsystems area are normally created or changed by dSPACE tools. They should not be modified by a design engineer or the administrator.

Typical combination of access rights

These are typical combinations of access rights:

rwrw	Read/write permission for all.
rwr–	The user has no write permission. Note that, in this case the user is allowed to modify the access rights and enable his/her own write permission (*rwrw*).
r–r–	The user and administrator have no write permission. Note that, the user is not able to modify the rights to *rwrw*, because he/she is not allowed to set the admin rights.

Access modes

To have access to protected DD objects, you must switch to the corresponding admin or user access mode in the Data Dictionary Manager or the DD MATLAB API.

Admin access mode Administrators work in the admin access mode, where the development environment for all users can be configured and the standardized variables are managed. The admin access rights include the user access rights. The admin access mode can be protected by a password (see below).

User access mode Design engineers work in the user access mode. They work with write-protected common objects maintained by the administrator, which they cannot modify. They can create new objects, e.g., new variables. The user access mode cannot be protected by a password.

For more information, refer to How to Change Access Rights on page 49 and Working with Access Rights on page 133.

Password protection

The admin access mode can be password-protected. The encrypted password is stored in the DD project file. If your DD project file includes password-protected included DD files, you must enter the password of each included DD file if it is different from the current admin password when you load the file.

Related topics

Basics

HowTos

Data Model of the TargetLink Data Dictionary

Introduction

The **TargetLink Data Dictionary** is able to store any tree of DD objects with arbitrary properties. The tools working with the **TargetLink Data Dictionary** need some specific objects with clearly defined properties. The Data Model defines which kinds of DD objects exist and which properties are defined for them.

Where to go from here

Information in this section

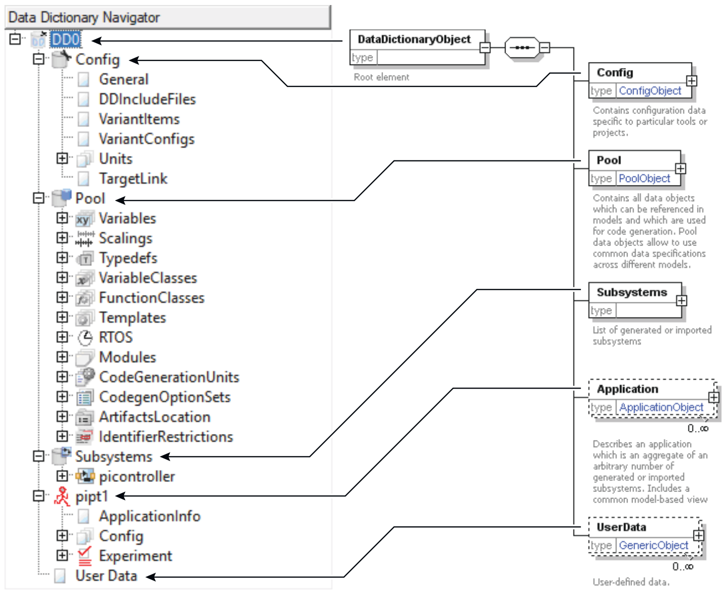
Basics on the Data Model as Shown in the DD Object Tree

Objects defined in the TargetLink Data Dictionary

The Data Model defines the DD objects, including their properties, that are allowed in the **TargetLink Data Dictionary**. Additionally, it defines the dependencies between the DD objects. Some DD objects can have child DD objects which can have their own child DD objects. This builds the structure for all DD objects contained in the **TargetLink Data Dictionary**. The structure is visible in the DD object tree, see Basics on the Data Dictionary Structure (DD Object Tree) on page 17.

Example

The following example shows an extract of the Data Model and the corresponding part of the DD object tree.



Contains configuration data specific to particular tools or projects.

Contains all data objects which can be referenced in models and which are used for code generation. Pool data objects allow to use common data specifications across different models.

List of generated or imported subsystems

Describes an application which is an aggregate of an arbitrary number of generated or imported subsystems. Includes a common model-based view

User-defined data.

The DataDictionaryObject has 3 mandatory children (Config, Pool, Subsystems) and optionally an arbitrary number of Application and/or UserData objects. The hierarchy is shown by the DD object tree.

Related topics

Basics

Basics on the Data Dictionary Structure (DD Object Tree).. 17

Overview of Commonly Used DD Objects

Most important object kinds

The TargetLink Data Dictionary contains numerous object kinds to describe data relevant for ECU applications. The DD objects are defined in the data model.

The following table shows the most important kinds of DD objects used by TargetLink:

Object	**Description**
Variable	Defines the properties of a variable in the C code. All kinds of variables are supported: plain variables, pointers, macros, structs, and bitfields. The most important variable properties are Type, Class, Scaling, Value, Min, Max, Address, NameTemplate. Other properties describe how a variable can be accessed in a calibration system, for example, ByteOrder, Deposit, Format, and MaxRefreshRate, for example. Variable objects are located in `/Pool/Variables` and `/Subsystems/<subsystem>/<module>/Variables`.
Typedef	Defines a C data type, including pointers, structs and bitfields. You can create your own user-defined types. Each typedef is mapped to one of the base types, such as Bool, Int8, or Float64. If required, you can also associate a typedef with constraints, such as a scaling formula or min/max values. Typedef objects are located in `/Pool/Typedefs` and `/Subsystems/<subsystems>/<module>/Typedefs`.
Scaling	Describes how to convert the fixed-point representation of a variable in the ECU memory to the corresponding physical value. Calibration systems use the scaling to read or write ECU variables. Scaling objects are located in `/Pool/Scalings` and `/Subsystems/<subsystems>/Scalings`. They can also be attached as child objects of an individual Variable object.
VariableClass	Defines the scope of variables, their storage in the ECU memory and their declaration and initialization in the generated C code. VariableClass objects are located in `/Pool/VariableClasses` and `/Subsystems/<subsystem>/VariableClasses`.
FunctionClass	Defines the scope of functions, their storage in the ECU memory, and their declaration in the generated C code, e.g., `GLOBAL_FCN,` `STATIC_FCN`, or `INTERRUPT_FCN`. FunctionClass objects are located in `/Pool/FunctionClasses` and `/Subsystems/<subsystem>/FunctionClasses`.
Subsystem	Describes a part of the application that results from one Code Generator run. Subsystem objects mainly contain a list of generated C modules and a ModelView, which links the blocks in the model with the generated code. A complete application can consist of several subsystems. Subsystems can also be created by import modules, for example, by an ASAM MCD-2 MC import module. Each subsystem is self-contained, i.e., there are no references to Scaling objects, Typedef objects, or VariableClass objects outside the subsystem. Therefore, you can change objects in the Pool area without invalidating the existing subsystem descriptions. Subsystem objects are located in `/Subsystems`.

Object	Description
Module	Describes a complete C module, including imported and exported variables and functions. The object also contains information on the files on which the module depends,, e.g., included C and header files. Each module belongs to a subsystem. Module objects are located in `/Subsystems/<subsystem>`.
Function	Describes a function in the C code, including a function interface description, a list of imported variables, and references of the Simulink blocks that implement the function in the model. A reused function can be called several times in the code. Therefore, each function instance is described in a separate child object referencing the task that calls the function instance. Function objects are located in `/Subsystems/<subsystems>/<module>`.
Application	Contains information on an ECU application, e.g., a list of the application's subsystems. Some experiment data is also stored in the Application object, such as the list of log variables and results of code coverage tests. Application objects are child objects of the DD root object.
Build	Contains information on an executable image of the application that was created by compiling and linking the associated subsystems. There can be an arbitrary number of builds for a single application, e.g., one build for an evaluation board (physical or virtual) and another build for the host PC. The Build object also contains information about base data types on the target and symbol table information. Build objects are located in `/<Application>`.
IdentifierRestriction	Restricts identifiers so that they are treated as keywords, type identifiers, or value identifiers. Lets you configure the message emitted on constraint violation. IdentifierRestriction objects are located in `/Pool/IdentifierRestrictions`.
Autosar	Contains DD objects that describe AUTOSAR-compatible software systems. The Autosar object is located in `/Pool/Autosar`.
CodegenOptionSet	Describes a set of TargetLink Code Generator options. As global switches, these options influence the code generation of TargetLink. CodegenOptionSet objects are located in `/Pool/CodegenOptionSets`.

Basics on Accessing DD Objects via Paths and Handles

DD object paths

DD objects can also be identified by their paths in the DD object tree. Absolute DD object paths begin with a slash, i.e., they are relative to the DD root object, for example, `/Pool/Variables/MyVar`. If you use multiple DD workspaces, you have to prefix the path with the path to the workspace's root object, for example, `//DD4//Pool/Variables/MyVar`.

For further information, refer to Object Paths.

Reference properties

Reference Properties are properties which point to objects. For example, if a variable in the Pool area references a Scaling object, the corresponding DD context object is `/Pool/Scalings`. This means that the reference `MyScalings/SC_Speed` evaluates the DD object `/Pool/Scalings/MyScalings/SC_Speed`. For a variable in the `Subsystems` area, the corresponding DD context object for scalings is `/Subsystems/<subsystems>/Scalings`.

Tip

You can get the DD context object via MATLAB API:

```
[hDDObject,errorCode] =
dsdd('GetContextObject',<objectIdentifier>[,<propertyName>]);
```

DD handle

The Data Dictionary Manager supports quick access to DD objects by entering DD object handles (DD handles). Each DD object is identified by a unique handle which is an unsigned integer. You can simply enter or paste the handle number into the DD object list. You can use this feature if you develop and debug M-scripts and need quick access to an object in the Data Dictionary Manager. However, DD handles are not persistent and expire if you delete their objects. If an object is recreated after having been deleted, it is assigned to another handle value. The handle also expires if you close the DD or the MATLAB session.

You can get the handle number in different ways. For example,

- by using the `GetDDAttributes` and `GetAttribute` DD MATLAB API functions.
- by using the Show Object Details context command in the DD Navigator pane.
- by reading the status bar at the bottom of the Data Dictionary Manager.

For information on the DD object list, refer to Overview of the User Interface on page 39.

For information on the DD MATLAB API and the available commands, refer to TargetLink Data Dictionary Reference.

Reference properties: Transforming specified DD handles into specified reference paths

DD objects often contain properties with reference paths to other DD objects. However, DD MATLAB API functions sometimes return handles instead of reference paths to DD objects. Using the `HandleRefsToStringRefs` API function lets you easily transform handles into reference paths instead of transforming each reference property manually. It converts all handle reference properties of all objects in a subtree. Empty reference properties are left untouched. This also applies to reference properties with more than one variant of which at least one is empty.

For information (*including detailed examples*) on how to use the `HandleRefsToStringRefs` command, refer to HandleRefsToStringRefs.

DD environment variables

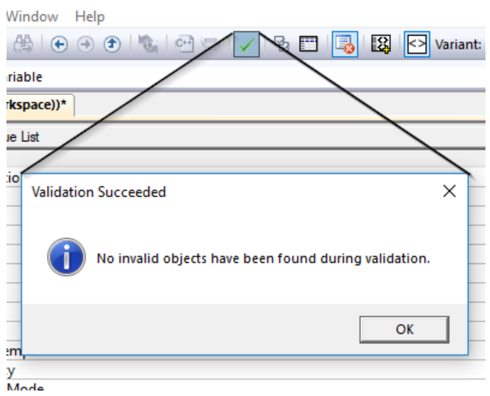
DD path names can contain DD environment variables enclosed by $ signs. When a DD path is evaluated, the environment variables are expanded to their current values. For example, if the environment variable `BR` is set to `BR42`, then the path name specification `CAL/$BR$/Kp` expands to `CAL/BR42/Kp`.

For information on how to use DD environment variables, refer to Basics on DD Environment Variables on page 132.

Basics on Validating Data Dictionaries

Compliance with the data model

For data exchange in the tool chain it is important that the objects in your data dictionary do not violate the Data Model. There are mechanisms which ensure that you cannot set invalid properties with the Data Dictionary Manager or the DD MATLAB API. Nevertheless, there are cross-dependencies between properties and objects which might result in invalid states. To check if your data dictionary is valid with respect to the data model, you can call a validation routine from the Data Dictionary Manager, which reports possible problems. Alternatively, you can use the DD MATLAB API for DD object validation.



Validation Succeeded

No invalid objects have been found during validation.

Related topics

HowTos

How to Validate DD Objects.. 80

References

Validate (📖 TargetLink Data Dictionary Manager Reference)

Managing Data with the Data Dictionary Manager

Introduction

The TargetLink Data Dictionary Manager is the central user interface of the TargetLink Data Dictionary (DD). It displays DD objects in a tree structure and gives you an easy way to handle DD objects and their properties.

Where to go from here

Information in this section

Getting Started with the Data Dictionary Manager

Introduction

The **TargetLink Data Dictionary Manager** provides some general functions to configure its view and behavior. You can navigate through the DD object tree via different methods. You can change the user mode to access DD objects protected by different access rights.

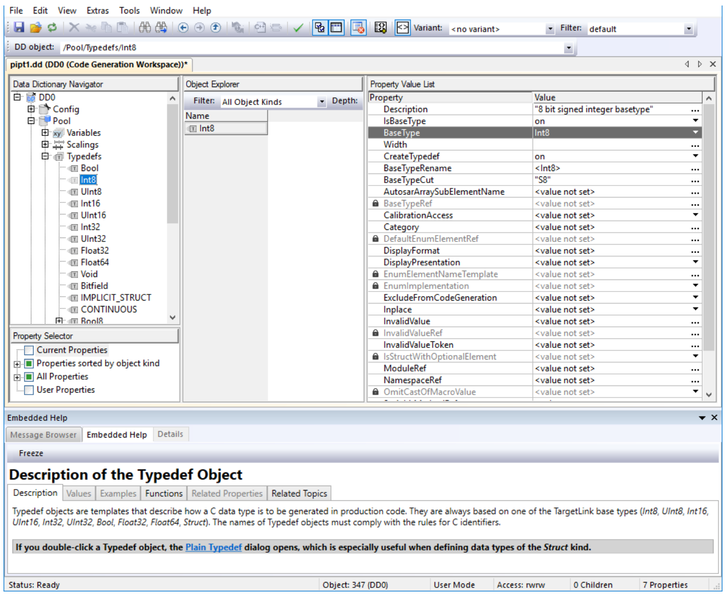
Where to go from here

Information in this section

Basics on the Data Dictionary Manager

Central user interface of the Data Dictionary

The **Data Dictionary Manager** is the central graphical user interface of the **Data Dictionary** for accessing the DD object tree.



DD object: /Pool/Typedefs/Int8

Variant: <no variant> Filter: default

Property	Value
Description	"8 bit signed integer basetype"
IsBaseType	on
BaseType	Int8
Width	
CreateTypedef	on
BaseTypeRename	<Int8>
BaseTypeCut	"S8"
AutosarArraySubElementName	<value not set>
BaseTypeRef	<value not set>
CalibrationAccess	<value not set>
Category	<value not set>
DefaultEnumElementRef	<value not set>
DisplayFormat	<value not set>
DisplayPresentation	<value not set>
EnumElementNameTemplate	<value not set>
EnumImplementation	<value not set>
ExcludeFromCodeGeneration	<value not set>
Inplace	<value not set>
InvalidValue	<value not set>
InvalidValueRef	<value not set>
InvalidValueToken	<value not set>
IsStructWithOptionalElement	<value not set>
ModuleRef	<value not set>
NamespaceRef	<value not set>
OmitCastOfMacroValue	<value not set>

Description of the Typedef Object

Typedef objects are templates that describe how a C data type is to be generated in production code. They are always based on one of the TargetLink base types (*Int8, UInt8, Int16, UInt16, Int32, UInt32, Bool, Float32, Float64, Struct*). The names of Typedef objects must comply with the rules for C identifiers.

If you double-click a Typedef object, the Plain Typedef dialog opens, which is especially useful when defining data types of the *Struct* kind.

Status: Ready | Object: 347 (DD0) | User Mode | Access: rwrw | 0 Children | 7 Properties

DD objects

You can define various kinds of DD objects. The available DD objects are defined in the DD data model. Refer to Data Model of the TargetLink Data Dictionary on page 21.

DD object tree

The DD object tree lets you organize the DD objects for a clear overview. The DD object tree is located in the **Data Dictionary Navigator**.

DD project files

DD project files let you archive all data of a DD workspace in one step or make a copy of it for another user. Refer to:

- How to Open DD Project Files in DD Workspaces on page 60
- How to Save DD Workspaces to DD Project Files on page 62

You can associate an existing DD project file with a model. This lets you use the same DD project file for different models. Refer to:

- How to Associate a DD Project File with a TargetLink Model on page 60
- How to Associate a DD Project File with a TargetLink Model via API on page 121

DD workspace

If you create a new DD workspace, it is available only in memory at first and you have to save it to a DD project file. The **Data Dictionary** supports any number of DD workspaces.

If you create a new DD workspace file, you can select a suitable DD template. Refer to:

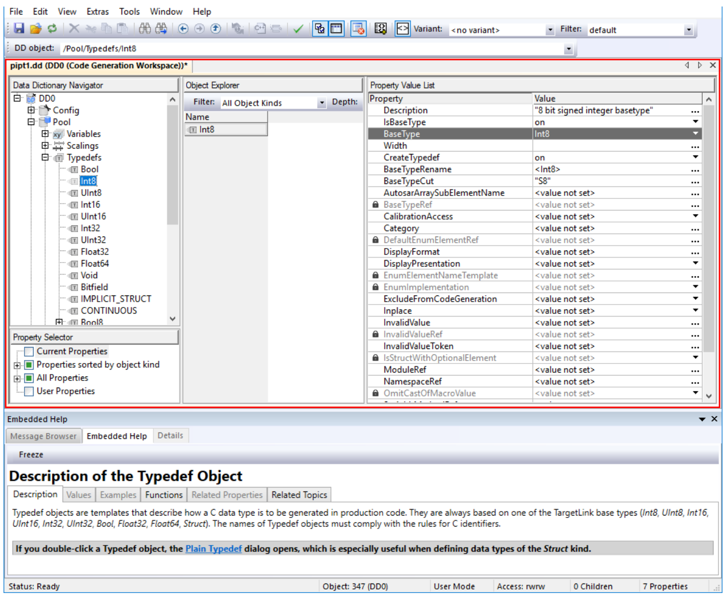
- How to Create DD Workspaces on page 58
- New - Create New DD Workspace (TargetLink Data Dictionary Manager Reference)

Each DD workspace can be represented by a DD workspace pane that contains the following panes:

- Data Dictionary Navigator
- Property Selector
- Object Explorer
- Property Value List

The following panes are displayed separately:

- Message Browser
- Embedded Help
- Details



Property	Value
Description	"8 bit signed integer basetype"
IsBaseType	on
BaseType	Int8
Width	
CreateTypedef	on
BaseTypeRename	<Int8>
BaseTypeCut	"S8"
AutosarArraySubElementName	<value not set>
BaseTypeRef	<value not set>
CalibrationAccess	<value not set>
Category	<value not set>
DefaultEnumElementRef	<value not set>
DisplayFormat	<value not set>
DisplayPresentation	<value not set>
EnumElementNameTemplate	<value not set>
EnumImplementation	<value not set>
ExcludeFromCodeGeneration	<value not set>
Inplace	<value not set>
InvalidValue	<value not set>
InvalidValueRef	<value not set>
InvalidValueToken	<value not set>
IsStructWithOptionalElement	<value not set>
ModuleRef	<value not set>
NamespaceRef	<value not set>
OmitCastOfMacroValue	<value not set>

Description of the Typedef Object

Description | Values | Examples | Functions | Related Properties | Related Topics

Typedef objects are templates that describe how a C data type is to be generated in production code. They are always based on one of the TargetLink base types (*Int8, UInt8, Int16, UInt16, Int32, UInt32, Bool, Float32, Float64, Struct*). The names of Typedef objects must comply with the rules for C identifiers.

If you double-click a Typedef object, the Plain Typedef dialog opens, which is especially useful when defining data types of the *Struct* kind.

Status: Ready | Object: 347 (DD0) | User Mode | Access: rwrw | 0 Children | 7 Properties

DD workspaces are categorized as follows:

- Code Generation Workspace
- Secondary workspace
- Reserved workspace

The **Code Generation Workspace** DD0 is the default DD workspace and used for model parameterization and code generation. The secondary workspace can be used to visualize partial DDs and inspect or compare previous versions of the

DD. The reserved workspaces are reserved entities and internally used by TargetLink.

The default Workspace Name of a new DD workspace is `DD<WorkspaceNumber>`. To change the name of an existing DD workspace, you can use the **`SetAttribute`** API command:

```
dsdd('SetDDAttribute', <WorkspaceID>, name, <DescriptiveName>);
```

To create a new DD workspace and give it a custom name, you can use the **`CreateDD`** API command:

```
dsdd('CreateDD', 'DDIdx', <WorkspaceID>, 'name', <DescriptiveName>);
```

To create a new DD workspace using a template DD, you can use the **`New`** API command:

```
dsdd('New', 'DDIdx', <WorkspaceID>, 'Template', <TemplateFileName>,
'name', <DescriptiveName>)
```

Unless you load a DD project file, each DD workspace is associated with a unique file name called `untitled_DD<WorkspaceID>.dd`. You can save the open DD workspace a file via the Save As menu command. Edit the name as required.

The tab of each DD workspace pane can display the following information:

- The name of the file associated with the DD workspace.
- The name of the workspace.
- The custom name assigned to the workspace.

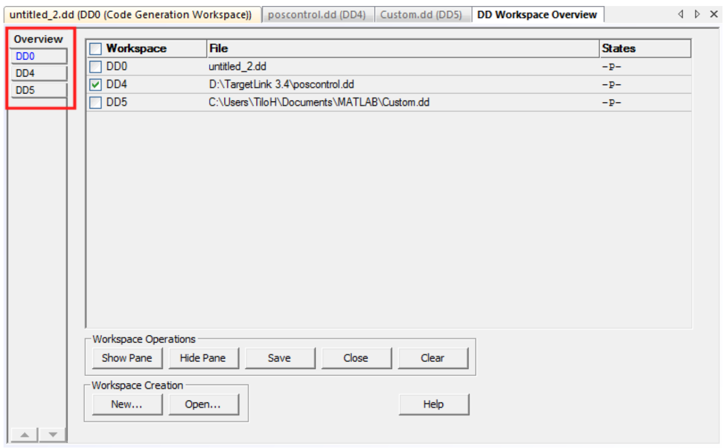
The name of the active DD workspace is displayed in bold in the title bar of the DD workspace pane and in the title bar of the Data Dictionary Manager. Actions such as keyboard entries, menu or toolbar commands apply to this pane.

Managing DD workspaces

All existing DD workspaces, even those without an associated DD project file, are listed in the submenu of this command. You can select one to open it.

The DD Workspace Overview displays all the DD workspaces, even existing DD workspaces that are not displayed in a separate pane yet or that do not have an associated DD project file. You can browse through the DD workspaces to inspect their properties and perform operations on them.

The DD Workspace Overview pane consists of the Overview pane and a separate pane for each existing DD workspace, for example, DD0, DD4, and DD5. To inspect the properties of a DD workspace, click its representation in the Overview pane.



Workspace	File	States
☐ DD0	untitled_2.dd	-P-
✓ DD4	D:\TargetLink 3.4\poscontrol.dd	-P-
☐ DD5	C:\Users\TiloH\Documents\MATLAB\Custom.dd	-P-

The DD Workspace Overview pane lets you execute various commands for DD workspace creation and operation. The commands are executed for the selected DD workspaces simultaneously (multiple selection). In contrast, each individual DD workspace pane provides detailed information on the selected DD workspace and functions for single DD workspace operations. Refer to DD Workspace Overview (TargetLink Data Dictionary Manager Reference).

File import and export

You can import or export files in different formats to or from the Data Dictionary Manager. Refer to:

- How to Import Data Files to DD Workspaces on page 50
- How to Export Data Files from DD Workspaces on page 51

Access modes

You can switch between the following Data Dictionary Manager modes to control access to the DD objects:

- Administrator mode
- User mode

Refer to How to Switch User Modes on page 48.

Working with DD objects

The Data Dictionary Manager provides all functions required for working with DD objects. Refer to:

- How to Create New Data Dictionary Objects on page 73
- How to Set Properties of Data Dictionary Objects on page 75
- How to Rename Data Dictionary Objects on page 77
- How to Delete Data Dictionary Objects on page 78
- How to Validate DD Objects on page 80

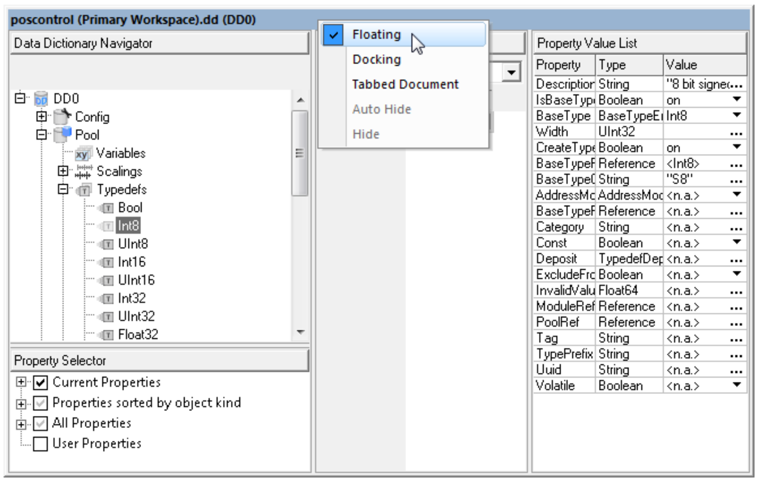
You can also use the DD MATLAB API to work with DD objects. Refer to:

- Basics on the DD MATLAB® API on page 116
- Basics on Handling Data Dictionary Objects via API on page 117

Customizing the screen layout

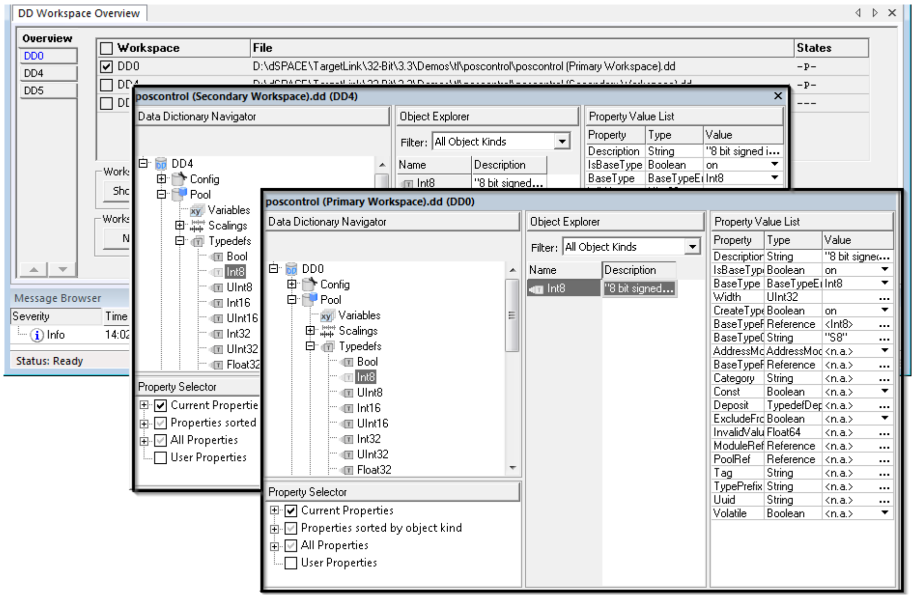
The screen layout defines which panes are displayed and where. When you open the **Data Dictionary Manager**, the default layout opens, displaying only the DD workspace pane of the DD project file associated with the model.

If you have further DD workspaces, you can use the multiple document interface (MDI) to display each DD workspace pane as a floating, docking, or tabbed document. You can access these commands on the context menu of the respective DD workspace pane or via drag & drop.



Property	Type	Value
Descriptior	String	"8 bit signec...
IsBaseTyp	Boolean	on
BaseType	BaseTypeE	Int8
Width	UInt32	...
CreateType	Boolean	on
BaseTypeF	Reference	<Int8> ...
BaseType(	String	"S8" ...
AddressMc	AddressMoc	<n.a.>
BaseTypeF	Reference	<n.a.> ...
Category	String	<n.a.> ...
Const	Boolean	<n.a.>
Deposit	TypedefDep	<n.a.> ...
ExcludeFrc	Boolean	<n.a.>
InvalidValu	Float64	<n.a.> ...
ModuleRef	Reference	<n.a.> ...
PoolRef	Reference	<n.a.> ...
Tag	String	<n.a.> ...
TypePrefix	String	<n.a.> ...
Uuid	String	<n.a.> ...
Volatile	Boolean	<n.a.>

The MDI display mechanisms allows you to work with multiple DD workspace panes simultaneously. As a result, you can easily perform operations such as dragging DD objects between different DD workspace panes. This is especially useful between floating DD workspace panes, refer to the following illustration. If you use the mouse to customize your screen display, the docking state of the pane automatically changes to **Floating** and the screen displays docking stickers that you can use to specify the new location in the **Data Dictionary Manager**.



DD Workspace Overview

Workspace	File	States
☑ DD0	D:\dSPACE\TargetLink\32-Bit\3.3\Demos\tl\poscontrol\poscontrol (Primary Workspace).dd	-p-

Message Browser

Severity	Time
Info	14:02

Status: Ready

poscontrol (Primary Workspace).dd (DD0)

Data Dictionary Navigator

```
DD0
  Config
  Pool
    Variables
    Scalings
    Typedefs
      Bool
      Int8
      UInt8
      Int16
      UInt16
      Int32
      UInt32
      Float32
```

Property Selector

- ☑ Current Properties
- ☑ Properties sorted by object kind
- ☑ All Properties
- ☐ User Properties

Object Explorer

Filter: All Object Kinds

Name	Description
Int8	"8 bit signed...

Property Value List

Property	Type	Value
Descriptior	String	"8 bit signe...
IsBaseTyp	Boolean	on
BaseType	BaseTypeE	Int8
Width	UInt32	
CreateTyp	Boolean	on
BaseTypeF	Reference	<Int8>
BaseType(	String	"S8"
AddressMc	AddressMoc	<n.a.>
BaseTypeF	Reference	<n.a.>
Category	String	<n.a.>
Const	Boolean	<n.a.>
Deposit	TypedefDep	<n.a.>
ExcludeFrc	Boolean	<n.a.>
InvalidValu	Float64	<n.a.>
ModuleRef	Reference	<n.a.>
PoolRef	Reference	<n.a.>
Tag	String	<n.a.>
TypePrefix	String	<n.a.>
Uuid	String	<n.a.>
Volatile	Boolean	<n.a.>

Navigating in the DD object tree

The Data Dictionary Manager offers several methods to navigate in the DD object tree and find DD objects. Refer to:

- How to Navigate in DD Object Trees on page 45
- How to Find Data Dictionary Objects on page 79

Filtering data

If the DD object tree contains a large number of DD objects, it is useful to define filtered user views of the DD object tree. Refer to Basics on Filter Rule Sets for the Data Model on page 121.

Sharing DD files

If DD objects are used in multiple DD project files, use included DD files to share these objects. This enables individual members of a development team to load their data from the same included DD file, which helps keep the data consistent. Refer to How to Include Partial Data Dictionary Files on page 65.

Variant handling of DD objects

If you need alternative values (variants) for the properties of a Variable object, you can specify code variants, data variants, or define a set of variable widths used for width-invariant code generation in the Data Dictionary.

Adding custom functionality to the Data Dictionary Manager

The Data Dictionary lets you add custom functionality to the Data Dictionary Manager. You can extend menus with user-defined menu items that call user-defined MATLAB functions, for example, M files. Refer to Adding Custom Functionality to the Data Dictionary Manager on page 135.

In addition, you can issue custom messages in the Message Browser or in custom output views. Refer to:

- How to Create Custom Output Views on page 146
- How to Display Custom Messages on page 147

Related topics

Basics

HowTos

References

CreateDD (📖 TargetLink Data Dictionary Reference)
DD Workspace Overview (📖 TargetLink Data Dictionary Manager Reference)
New - Create New DD Workspace (📖 TargetLink Data Dictionary Manager Reference)
Save As (📖 TargetLink Data Dictionary Manager Reference)
SetAttribute (📖 TargetLink Data Dictionary Reference)

How to Start the Data Dictionary Manager

Objective

The **TargetLink Data Dictionary Manager** can be started via MATLAB or in stand-alone mode (i.e., independently of MATLAB). If you open the **TargetLink Data Dictionary Manager** in the stand-alone mode, some utilities are not available, for example, menu extensions.

DD project file

When you open a Simulink model, the associated DD project file is automatically loaded by TargetLink. You can also open a DD project file with the TargetLink Data Dictionary Manager or the MATLAB API. In large applications, the same DD project file can be used by more than one model. A DD project file may contain other included DD files. For more information on included DD files, refer to How to Include Partial Data Dictionary Files on page 65.

Possible methods

- You can start the Data Dictionary Manager in full-featured mode. Refer to Method 1.
- You can start the Data Dictionary Manager in stand-alone mode. Refer to Method 2.

Method 1

To start the TargetLink Data Dictionary Manager in full-featured mode

1 Start MATLAB.

2 In the MATLAB Command Window, enter **`dsddman`**.

Method 2

To start the TargetLink Data Dictionary Manager in stand-alone mode

1 From the Start – All Programs – dSPACE TargetLink <version> - TargetLink Data Dictionary Manager.

Result

The TargetLink Data Dictionary Manager opens.

Tip

You can open the TargetLink Data Dictionary Manager with a specified DD project file using the command in the MATLAB Command Window or in an M file:

```
dsddman('Open', <ddProjectFile>)
```

Related topics

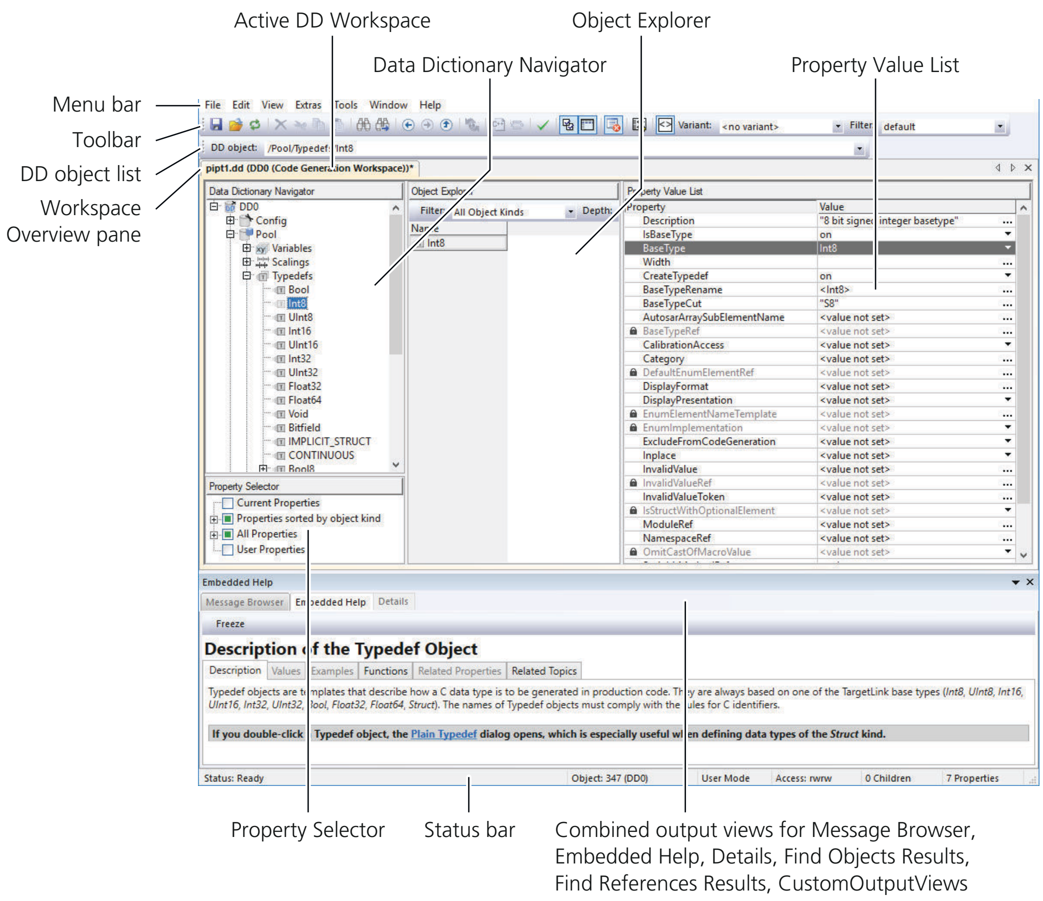
References

Data Dictionary Limitations (TargetLink Limitation Reference)
dsddman (TargetLink API Reference)

Overview of the User Interface

Main window

The main window of the **TargetLink Data Dictionary Manager** contains the following elements:



Property Value List

Property	Value
Description	"8 bit signed integer basetype"
IsBaseType	on
BaseType	Int8
Width	
CreateTypedef	on
BaseTypeRename	<Int8>
BaseTypeCut	"S8"
AutosarArraySubElementName	<value not set>
BaseTypeRef	<value not set>
CalibrationAccess	<value not set>
Category	<value not set>
DefaultEnumElementRef	<value not set>
DisplayFormat	<value not set>
DisplayPresentation	<value not set>
EnumElementNameTemplate	<value not set>
EnumImplementation	<value not set>
ExcludeFromCodeGeneration	<value not set>
Inplace	<value not set>
InvalidValue	<value not set>
InvalidValueRef	<value not set>
InvalidValueToken	<value not set>
IsStructWithOptionalElement	<value not set>
ModuleRef	<value not set>
NamespaceRef	<value not set>
OmitCastOfMacroValue	<value not set>

Description of the Typedef Object

Typedef objects are templates that describe how a C data type is to be generated in production code. They are always based on one of the TargetLink base types (*Int8*, *UInt8*, *Int16*, *UInt16*, *Int32*, *UInt32*, *Bool*, *Float32*, *Float64*, *Struct*). The names of Typedef objects must comply with the rules for C identifiers.

If you double-click a Typedef object, the Plain Typedef dialog opens, which is especially useful when defining data types of the *Struct* kind.

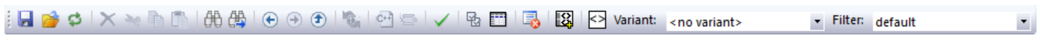
Status: Ready | Object: 347 (DD0) | User Mode | Access: rwrw | 0 Children | 7 Properties

Menu bar

The menu bar provides access to common functions and commands for working with the **TargetLink Data Dictionary Manager**. For more information, refer to the TargetLink Data Dictionary Manager Reference.

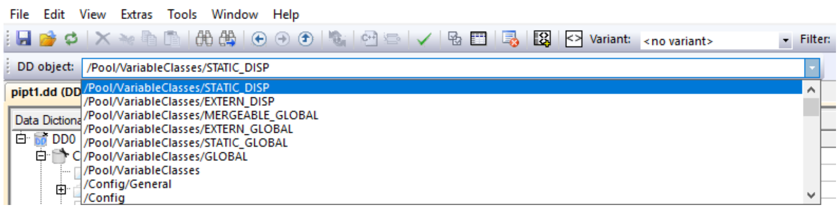
Toolbar

The toolbar provides quick access to frequently used commands. In addition, the **Filter** list lets you select user-defined filter rule sets to hide specific objects and properties.



DD object list

The **DD object** list displays the DD path of the currently selected DD object. Like the address bar in the File Explorer, you can access DD objects by typing or pasting a path (case-sensitive). The DD history list contains a history with the last selected DD objects as indicated in the screenshot below.



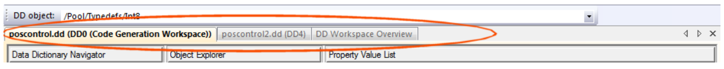
/Pool/VariableClasses/STATIC_DISP
/Pool/VariableClasses/STATIC_DISP
/Pool/VariableClasses/EXTERN_DISP
/Pool/VariableClasses/MERGEABLE_GLOBAL
/Pool/VariableClasses/EXTERN_GLOBAL
/Pool/VariableClasses/STATIC_GLOBAL
/Pool/VariableClasses/GLOBAL
/Pool/VariableClasses
/Config/General
/Config

Instead of a path, you can also enter a DD handle (unsigned integer). You can enter DD handles for any DD objects in any DD workspaces (not necessarily the currently selected). If entered, the DD Object Explorer automatically jumps to the DD workspace containing the referenced DD object.

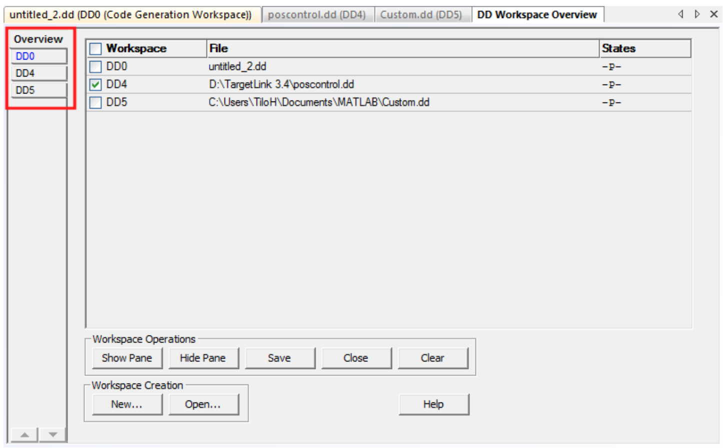
DD workspace

A DD workspace is an independent organizational unit (central data container) and the largest entity that can be saved to file or loaded from DD project file. It holds all the relevant information about an ECU application, for example, for code generation. The **TargetLink Data Dictionary** supports an arbitrary number of DD workspaces (DD 0 … DD n), each of which can be represented by a *DD Workspace pane* that contains a predefined, fixed set of DD panes, i.e., the **Data Dictionary Navigator**, the **Property Selector**, the **Object Explorer** and the **Property Value List**.

The **DD0** workspace is the **Code Generation Workspace**. It is the primary workspace, used for code generation. Its frame is highlighted.


DD object: /Pool/Typedefs/Int8
poscontrol.dd (DD0 (Code Generation Workspace))
poscontrol2.dd (DD4)
DD Workspace Overview
Data Dictionary Navigator
Object Explorer
Property Value List

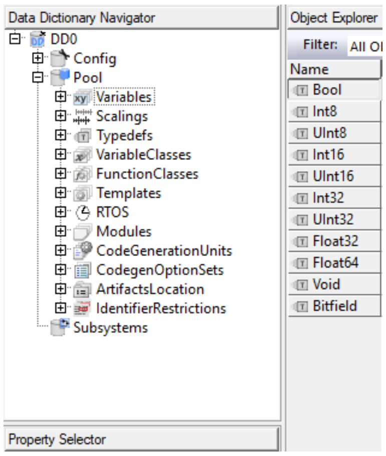
The **DD Workspace Overview** displays all the DD workspaces, even existing DD workspaces that are not displayed in a separate pane yet or that do not have an associated DD project file. You can browse through the DD workspaces to inspect their properties and perform operations on them.



Workspace	File	States
☐ DD0	untitled_2.dd	-p-
✓ DD4	D:\TargetLink 3.4\poscontrol.dd	-p-
☐ DD5	C:\Users\TiloH\Documents\MATLAB\Custom.dd	-p-

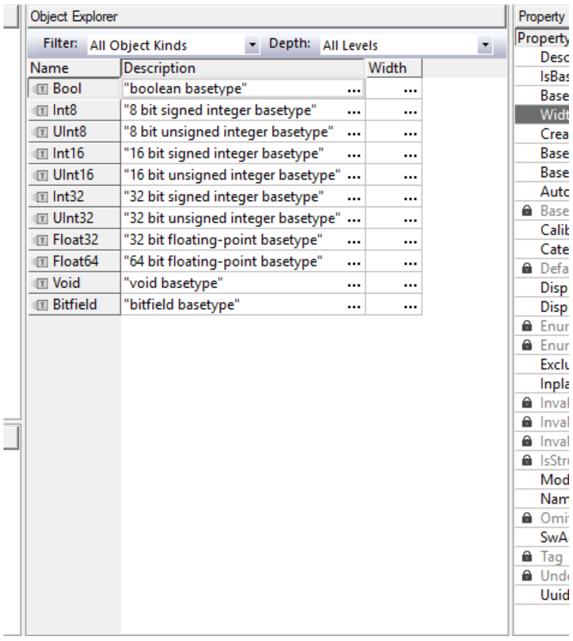
Data Dictionary Navigator

The Data Dictionary Navigator displays all DD objects in the DD object tree which belong to the currently open DD workspace.



Related panes for objects and properties

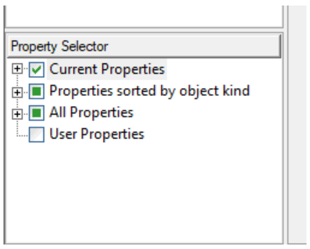
Object Explorer



Name	Description	Width
Bool	"boolean basetype" ...	...
Int8	"8 bit signed integer basetype" ...	...
UInt8	"8 bit unsigned integer basetype" ...	...
Int16	"16 bit signed integer basetype" ...	...
UInt16	"16 bit unsigned integer basetype" ...	...
Int32	"32 bit signed integer basetype" ...	...
UInt32	"32 bit unsigned integer basetype" ...	...
Float32	"32 bit floating-point basetype" ...	...
Float64	"64 bit floating-point basetype" ...	...
Void	"void basetype" ...	...
Bitfield	"bitfield basetype" ...	...

The Object Explorer displays DD objects and their properties in a table where you can display and edit the properties of multiple DD objects. You can open the Object Explorer in the View menu.

Property Selector

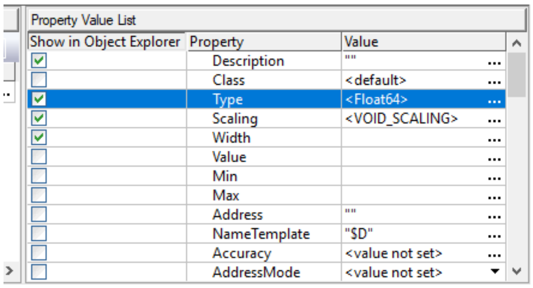


Property Selector

- ✓ Current Properties
- ▪ Properties sorted by object kind
- ▪ All Properties
- ☐ User Properties

The Property Selector lets you select the properties of the DD objects to be displayed in the Object Explorer. You can open the Property Selector in the View menu.

Property Value List



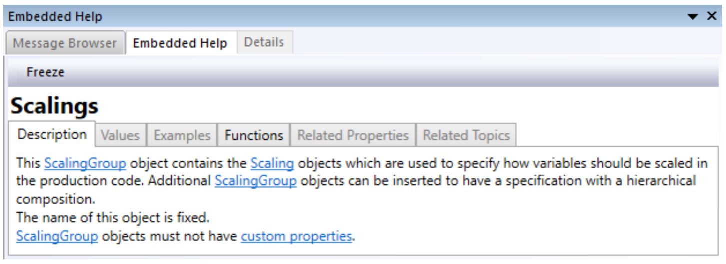
Property Value List

Show in Object Explorer	Property	Value
✓	Description	"" ...
☐	Class	<default> ...
✓	Type	<Float64> ...
✓	Scaling	<VOID_SCALING> ...
✓	Width	...
☐	Value	...
☐	Min	...
☐	Max	...
☐	Address	"" ...
☐	NameTemplate	"$D" ...
☐	Accuracy	<value not set> ...
☐	AddressMode	<value not set>

The Property Value List, which is permanently visible, displays and lets you edit the properties of the currently selected DD object.

Embedded Help pane

The Embedded Help pane provides detailed descriptions on selected DD objects or properties. It is activated by default. From the Help menu, click Show Embedded Help to close or open the pane.



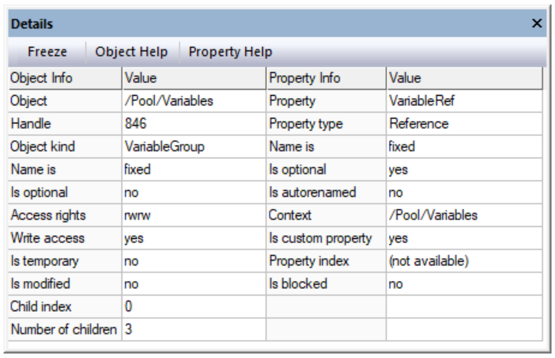
Scalings

This ScalingGroup object contains the Scaling objects which are used to specify how variables should be scaled in the production code. Additional ScalingGroup objects can be inserted to have a specification with a hierarchical composition.
The name of this object is fixed.
ScalingGroup objects must not have custom properties.

For further information, refer to Show Embedded Help (📖 TargetLink Data Dictionary Manager Reference).

Details pane

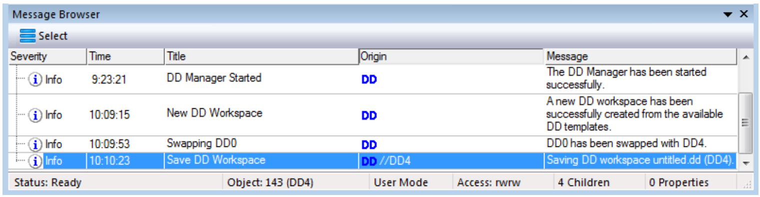
The Details pane displays information on the selected DD object and, if selected, its associated properties.



Object Info	Value	Property Info	Value
Object	/Pool/Variables	Property	VariableRef
Handle	846	Property type	Reference
Object kind	VariableGroup	Name is	fixed
Name is	fixed	Is optional	yes
Is optional	no	Is autorenamed	no
Access rights	rwrw	Context	/Pool/Variables
Write access	yes	Is custom property	yes
Is temporary	no	Property index	(not available)
Is modified	no	Is blocked	no
Child index	0		
Number of children	3		

Message Browser pane

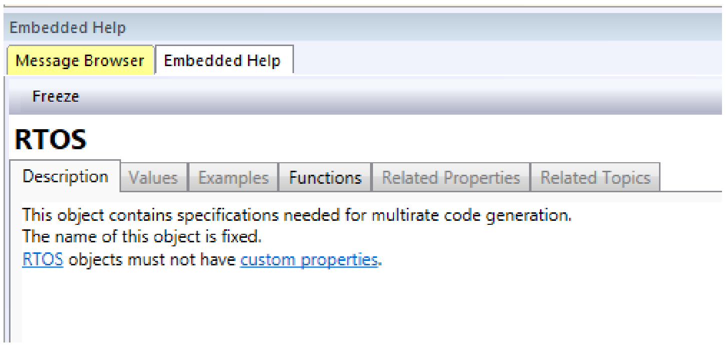
The Message Browser pane displays all the messages of the Data Dictionary Manager. You can open the Message Browser via View - Show Output View or via the toolbar button Show application Trace.



Severity	Time	Title	Origin	Message
Info	9:23:21	DD Manager Started	DD	The DD Manager has been started successfully.
Info	10:09:15	New DD Workspace	DD	A new DD workspace has been successfully created from the available DD templates.
Info	10:09:53	Swapping DD0	DD	DD0 has been swapped with DD4.
Info	10:10:23	Save DD Workspace	DD //DD4	Saving DD workspace untitled.dd (DD4).

Status: Ready | Object: 143 (DD4) | User Mode | Access: rwrw | 4 Children | 0 Properties

If the Message Browser pane is opened but hidden because of an active Embedded Help pane, the tab of the Message Browser indicates that there are new messages by changing the tab color.



RTOS

This object contains specifications needed for multirate code generation.
The name of this object is fixed.
RTOS objects must not have custom properties.

- A new Info message - blue tab
- A new Warning message - yellow tab
- A new Error message - red tab

Custom output views

Custom output views display custom messages that can be issued by your own scripts and tools.

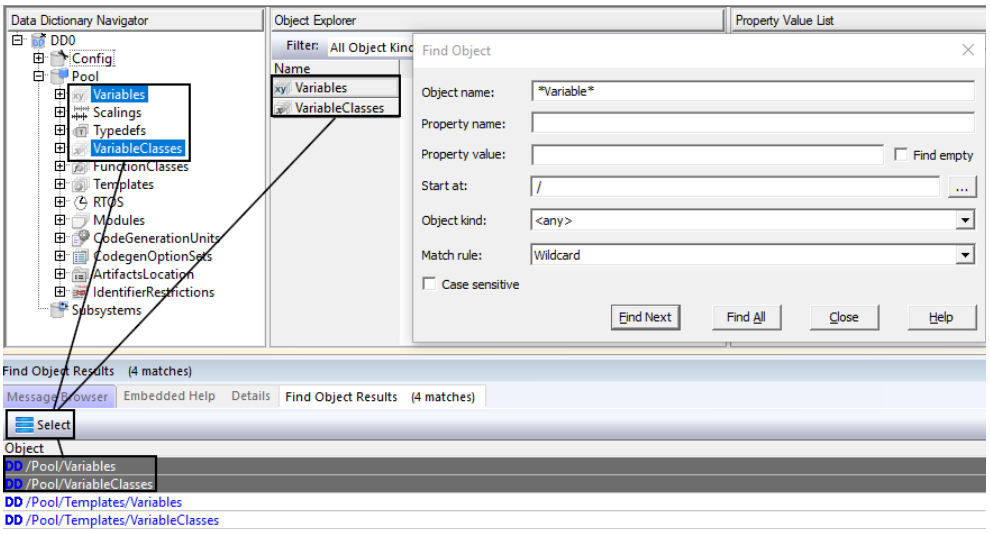
Find Object Results pane

The Find Object Results pane displays the objects that match the specified search criteria. It opens when the Find All command in the Find Object dialog is executed.

If you have multiple DD objects in the Find Object Results pane, you can multiselect them via left click and then click the Select button:

- The first object of the search result is selected and displayed in the Data Dictionary Navigator *and* in the Object Explorer.
- All other objects of the search result are selected in the Data Dictionary Navigator *and* in the Object Explorer.

Refer to the illustration below.



Object name: *Variable*
Property name:
Property value:
Start at: /
Object kind: <any>
Match rule: Wildcard

DD /Pool/Variables
DD /Pool/VariableClasses
DD /Pool/Templates/Variables
DD /Pool/Templates/VariableClasses

Find References Results pane

The Find References Results pane displays the objects that reference a selected DD object. It opens when the Find References command is executed on a DD object whose references you want to find.

Status bar

The status bar of the TargetLink Data Dictionary Manager displays general information, such as the status of the DD workspace and the user mode, and object-specific information, such as the object handle and the number of properties set for the currently selected DD object.

Related topics

HowTos

How to Create Custom Output Views.. 146
How to Open DD Project Files in DD Workspaces 60

How to Navigate in DD Object Trees

Objective

The Data Dictionary Manager makes it easy to navigate in the DD object tree.

Possible methods

When working with the Data Dictionary Navigator, you often have to navigate to different DD objects. The Data Dictionary Manager provides three different methods to do this:

- If you know the exact path or the DD handle of the DD object you want to navigate to, refer to Method 1.
- If a child DD object is selected and you want to navigate to its parent, refer to Method 2.
- The Data Dictionary Manager creates an object list containing a history of the selected DD objects when you navigate through the DD object tree. You can best navigate to DD objects by selecting a specific entry in the list. Refer to Method 3.

Method 1

To navigate to a specific DD object in the DD object tree

1 In the **DD object** list, enter the exact path (case-sensitive) or a DD handle (unsigned 32-bit number) of the DD object you want to navigate to.

The DD object is selected and its path is added to the history of the **DD object** list.

Tip

The **DD object** list contains the history of the selected DD objects. You can also select the path from this list.

Method 2

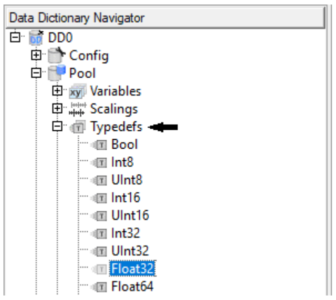
To navigate to a parent DD object of the currently selected DD object

1 In the toolbar, click .

The parent DD object of the selected child is selected.

Example

If the /Pool/`Typedefs/Float32` DD object is selected, clicking selects the /Pool/`TypeDefs` DD object which is Float32's parent DD object.



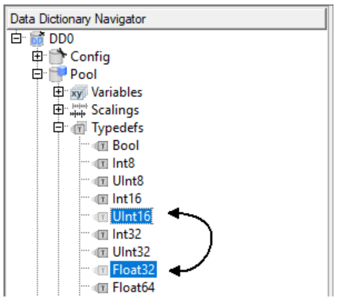
Method 3

To navigate to previously selected DD objects

1. If you want to navigate to a previously selected DD object, click the back button in the toolbar.
   The DD object that comes next in the history is selected.
2. If you want to navigate to a DD object you selected more recently, click the forward button in the toolbar.

Example

In this example you first selected the DD object `/Typedefs/UInt16` and afterwards the DD object `/Typedefs/Float32`. Using and changes the selection from one DD object to the other.



Tip

Alternatively, you can use the DD object list to navigate to previously selected DD objects.

Result

You have navigated to the desired DD object.

Related topics

HowTos

How to Get Help on DD Objects and Properties

Introduction

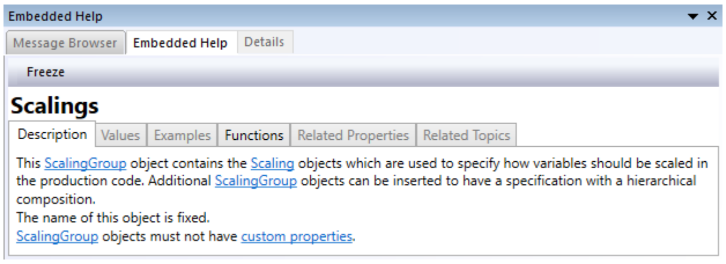
You can get information on all DD objects and their properties.

Method

To get help on DD objects (properties)

1 Select the DD object or property you want to get information on.

2
- By default, the **Embedded Help** pane provides detailed information on DD objects and properties of the Data Model. You can close or open the Embedded Help pane by clicking **Help -Show Embedded Help**.



This ScalingGroup object contains the Scaling objects which are used to specify how variables should be scaled in the production code. Additional ScalingGroup objects can be inserted to have a specification with a hierarchical composition.
The name of this object is fixed.
ScalingGroup objects must not have custom properties.

- Press **F1** to get additional help via dSPACE Help.
- A third method to get help is to select **Show Details** (TargetLink Data Dictionary Manager Reference) in the context menu of the DD object or property. As a result of this action the **Details pane** is displayed. It contains more information on the selected object or property. In the **Details pane**, use the **Object Help** or **Property Help** buttons to open dSPACE Help.

Result

The information on the DD object (property) you are interested in is displayed.

How to Switch User Modes

Objective

The access mode defines which DD objects are readable or writable. Different access rights can be assigned for each DD object, i.e., access rights are independent of the DD object tree.

Administrator and user mode

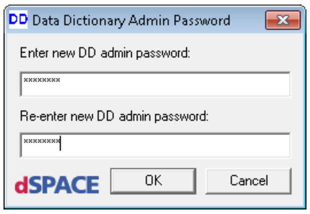
The Data Dictionary supports an administrator mode and a user mode. For further information, refer to Basics on User Modes and Access Rights on page 19.

Method

To change to the administrator mode or user mode

1 From the menu bar, select Extras — Change to Administrator/User Mode.

2 If you want to change to the administrator mode and a password is set, you have to enter password, which depends on the opened DD project file.


Data Dictionary Admin Password

Enter new DD admin password: ********

Re-enter new DD admin password: ********

Result

The Data Dictionary Manager changes the access mode. The currently set access mode is displayed in the status bar.

Related topics

Basics

Basics on User Modes and Access Rights.. 19

References

Change to Administrator/User Mode (📖 TargetLink Data Dictionary Manager Reference)
Set Admin Password (📖 TargetLink Data Dictionary Manager Reference)

How to Change Access Rights

Objective

To change the access rights of a DD object. Each DD object is associated with read and write access rights for the administrator (admin access mode) and the user (user access mode).

Note

Properties that have read-only access are indicated by a 🔒 symbol.

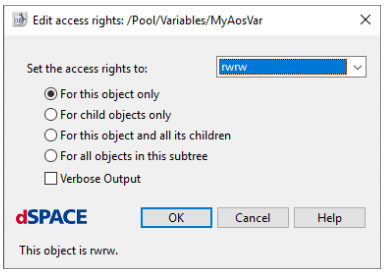
Method

To change access rights

1 In the DD object tree, select the DD object.

2 On the context menu, click Access.

3 In the Edit Access Rights dialog, set the access rights as required.



Edit access rights: /Pool/Variables/MyAosVar

Set the access rights to: rwrw

[✓ For this object only]
For child objects only
For this object and all its children
For all objects in this subtree

☐ Verbose Output

This object is rwrw.

Result

You changed the access rights.

Note

- Users cannot set the access rights for the administrator.
- Access rights are not automatically transferred from a parent to a newly created DD child object.

Related topics

Basics

Basics on User Modes and Access Rights.. 19

References

Access (📖 TargetLink Data Dictionary Manager Reference)

Managing Data Dictionary Files

Introduction

The TargetLink Data Dictionary saves the DD workspace in a DD project file. Files in other data formats can be imported to or exported from a DD workspace. If several members of your development team work with different DD project files, you can keep shared data consistent by using included DD files which are loaded into a subtree of the DD workspace.

Where to go from here

Information in this section

How to Import Data Files to DD Workspaces

Objective

To import data files to DD workspaces.

Supported file formats

The import interface of the Data Dictionary Manager is generic so that it can support different file formats. The supported file formats depend on your TargetLink license.

Method

To import data files to DD workspaces

1 From the menu bar of the Data Dictionary Manager, select File – Import.

All supported file formats are listed in the Import submenu.

2 Select the file format you want to import.

> **Note**
>
> OIL files can be imported only if DD0 is the active workspace.

The import pane for the selected file format opens.

3 Fill out the missing information and click Import.

Result

You imported data files to DD workspaces.

Related topics

Basics

Importing AUTOSAR Files (📖 TargetLink Interoperation and Exchange Guide)
Importing OIL Files (📖 TargetLink Interoperation and Exchange Guide)
Importing XML Files (📖 TargetLink Interoperation and Exchange Guide)

HowTos

How to Export Data Files from DD Workspaces.. 51

References

Import (📖 TargetLink Data Dictionary Manager Reference)

How to Export Data Files from DD Workspaces

Introduction

To export data files from DD workspaces.

Supported file formats

The export interface of the Data Dictionary Manager is generic so that it can support different file formats. The supported file formats depend on your TargetLink license.

Method

To export data files from DD workspaces

1 From the menu bar of the Data Dictionary Manager, select File – Export.
All supported file formats are listed in the Export submenu.

2 Select the file format you want to export.

Note

A2L files and OIL files can be exported only if DD0 is the active workspace.

The export pane for the selected file format opens.

3 Fill out the missing information and click **Export**.

Result

You exported data files from DD workspaces.

Related topics

Basics

Exchanging Data (TargetLink Interoperation and Exchange Guide)

HowTos

How to Import Data Files to DD Workspaces.. 50

References

Export (TargetLink Data Dictionary Manager Reference)

How to Swap a Secondary DD Workspace with the Primary DD Workspace

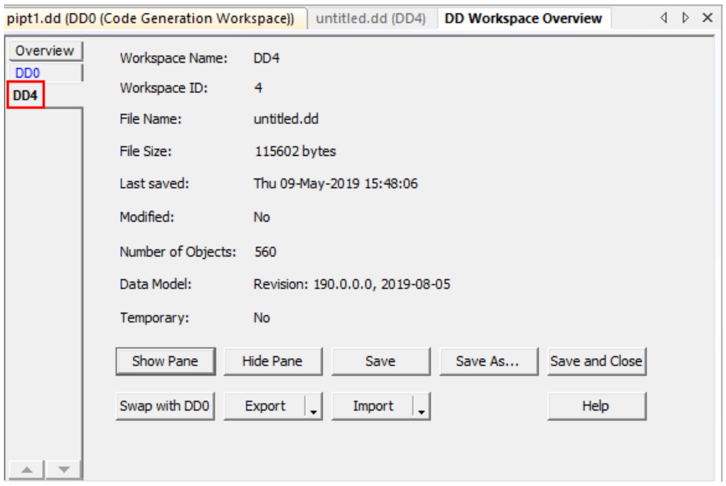
Objective

To swap a secondary DD workspace with the primary DD workspace (DD0) so that the secondary DD workspace becomes the primary DD workspace and is used for model parameterization and code generation.

Method

To swap a secondary DD workspace with the primary DD workspace

1 In the DD Workspace Overview pane of the Data Dictionary Manager, select the secondary DD workspace (here: DD4).



Workspace Name: DD4
Workspace ID: 4
File Name: untitled.dd
File Size: 115602 bytes
Last saved: Thu 09-May-2019 15:48:06
Modified: No
Number of Objects: 560
Data Model: Revision: 190.0.0.0, 2019-08-05
Temporary: No

2 Click the Swap with DD0 button.

Tip

As an alternative, you can click the Make Primary Workspace toolbar button ( ) in any DD workspace except for DD0 itself.

Result

You swapped a secondary DD workspace with the primary DD workspace to use it for model parameterization and code generation. The previous primary DD workspace has become the secondary workspace.

Note

Making another DD workspace the primary DD workspace does not affect settings in the current Simulink model. If the active DD workspace does not correspond to the model settings, warnings can occur on numerous actions, for example, before code generation.

Related topics

Basics

Basics on Opening and Handling DD Files

Introduction

The Data Dictionary Manager offers different ways of opening and handling DD files:

- You can open one or more DD files via the **Open** and **Load** commands.
- You can handle partial DD files via the **Point of Inclusion** dialog.

Commands to open DD files

In the Data Dictionary Manager, you can open DD files using one of the following commands:

Command	Description
Open	Open - Use Active DD Workspace (TargetLink Data Dictionary Manager Reference) Open - Create New DD Workspace (TargetLink Data Dictionary Manager Reference)
Load	Load (TargetLink Data Dictionary Manager Reference)

Note

You can load a DD file to any DD object, which might result in a Data Dictionary that does not comply with the DD data model, i.e., is invalid.

Automatic loading behavior

If you open a DD file by using the **Open** command, the loading behavior is as follows:

Type of Data Dictionary	Loading Behavior
Complete Data Dictionary	The active DD workspace is filled with the content of the DD file.
Partial Data Dictionary with metadata about its original position	The Data Dictionary Manager automatically reads the original DD subtree and loads the file contents to that position.
Partial Data Dictionary without metadata about its original position	The contents are loaded to a valid position according to the DD data model. If this is not possible, the contents are loaded to a `/tmp` DD subtree. You can manually move the objects to valid positions.

Note

Partial DD files saved with TargetLink versions prior to TargetLink 3.3 do not contain metadata.

Opening multiple DD files

You can open multiple DD files at once. However, only specific combinations of DD files are supported:

- You can multiselect DD files containing one complete Data Dictionary and one or more partial Data Dictionaries.
- You can multiselect DD files containing complete Data Dictionaries.
- You can multiselect files containing partial Data Dictionaries.

To open multiple DD files, use one of the following Open commands:

- Open - Use Current DD Workspace

DD file	Loading Behavior
DD project files	The content of the first selected DD project file is loaded to the current DD workspace and the others are loaded to new DD workspaces.
Partial DD files	The content of all selected partial DD files is loaded to the current DD workspace.

  Refer to Open - Use Active DD Workspace (TargetLink Data Dictionary Manager Reference).

- Open - Create New DD Workspace

DD file	Loading Behavior
DD project files	The content of each selected DD project file is loaded to a new separate DD workspace for each.
Partial DD files	The content of all selected partial DD files is loaded to the same new DD workspace.

  Refer to Open - Create New DD Workspace (TargetLink Data Dictionary Manager Reference).

Handling of inclusion points

Inclusion points allow you to reload a partial DD file to a specific position in the DD object tree. If you include a partial DD file in your DD workspace, it is called an included DD file. A typical use case for this mechanism is to access centrally managed configuration and pool data. You can let team members share the same configuration and pool data from the included DD file, which the administrator maintains on a network server.

Unlike data loaded by the Load command, data that is loaded via the included DD file mechanism is firmly referenced with the open DD project. Use the Point of Inclusion dialog to do the following:

- Include an externally saved partial DD file containing a DD subtree at a specific position in the DD workspace.

  Refer to How to Include Partial Data Dictionary Files on page 65.

- Extract an existing DD subtree from your DD workspace and save it to a newly created partial DD file.

  Refer to How to Separate DD Subtrees and Save Them as Included DD Files on page 68.

Related topics

Basics

HowTos

References

Open - Create New DD Workspace (📖 TargetLink Data Dictionary Manager Reference)
Open - Use Active DD Workspace (📖 TargetLink Data Dictionary Manager Reference)
Point of Inclusion (📖 TargetLink Data Dictionary Manager Reference)

How to Load DD Files

Objective

To load DD files to the active DD workspace.

Loading DD files

Unlike the Open command, the Load command does not clear the workspace before loading the selected DD file. Instead, the DD file is inserted in the selected DD object.

If you want to make changes to certain DD objects which where loaded to be automatically written back to the original DD file, you have to use Point of Inclusion (📖 TargetLink Data Dictionary Manager Reference).

Possible methods

Use one of the following methods to load a DD file to the active DD workspace:

- The contents of the DD file are merged into the selected DD object. Refer to Method 1 on page 56.
- The contents of the DD file become a DD child object of the selected DD object. Refer to Method 2 on page 57.

Method 1

To load a DD file into a DD object

1. In the context menu of the DD object to which you want to load the DD files, select Load.
2. In the Load DD file into /<path>/ dialog, select the DD file you want to load to the active DD workspace.
3. Clear the Load as child object checkbox.

4 Select a suitable Mode for handling conflicts between DD objects and properties in the active DD workspace and in the loaded DD file.

5 Click OK to load the selected DD file.

Method 2

To load a DD file as a DD child object

1 In the context menu of the DD object to which you want to load the DD files, select Load.

2 In the Load DD file into /<path>/ dialog, select the DD file you want to load to the active DD workspace.

3 Make sure that the Load as child object checkbox is selected.

4 Click OK to load the selected DD file.

Result

You loaded a DD file to the active DD workspace.

Note

It is recommended to validate the loaded DD objects, i.e., to check whether the newly loaded data complies with the DD data model. There is no automatic validation.

Related topics

Basics

Basics on the Data Dictionary Manager........ 31
Overview of the User Interface........ 39

HowTos

How to Merge or Replace DD Objects 92
How to Open DD Project Files in DD Workspaces 60
How to Save a DD Subtree to a Partial DD File........ 63

References

Load (📖 TargetLink Data Dictionary Manager Reference)
Load included Files (📖 TargetLink Data Dictionary Manager Reference)

How to Create DD Workspaces

Introduction

To create DD workspaces.

Predefined system templates

When you create a new DD workspace, you can either choose between 4 predefined system templates or select user-defined templates.

dsdd_master_basic.dd [System] Provides the new DD workspace with a basic set of predefined DD objects, such as DD Typedef, DD VariableClass, and DD FunctionClass objects.

dsdd_master_advanced.dd [System] Provides the new DD workspace with additional predefined objects, such as DD Template objects and DD RTOS objects required for multirate code generation.

dsdd_master_autosar4.dd [System] Provides the new DD workspace with additional predefined objects, such as DD Typedef and DD Variable objects to be used in AUTOSAR SWCs, and a DD object for specifying AUTOSAR software components and interfaces. Use this template to work with the AUTOSAR 4.x standard.

dsdd_master_adaptive_autosar.dd [System] Provides the new DD workspace with additional predefined objects, such as DD Typedef objects, to be used in AUTOSAR SWCs, and a DD object for specifying AUTOSAR software components and interfaces. Use this template to work with the Adaptive AUTOSAR standard.

If you created your own template and want to add it to the Predefined template list, the file must reside in a folder which is specified in the TargetLink Preferences. Refer to Topic Settings (TargetLink Tool and Utility Reference).

User-defined templates

You can use any DD project file as a user-defined template.

Tip

If you place a TXT file with the name of the DD project file you want to use as a user-defined template, TargetLink automatically adds the contained text to the Select DD Project File Template dialog as a template description.

Method

To create a DD workspace

1 From the menu bar of the Data Dictionary Manager, select File – New.

2 Select one of the following options:

- Click New – Use Active DD Workspace to clear the active DD workspace and open a DD project file in it. The DD workspace name, e.g., DD0, does not change.
- Click New - Create New DD Workspace to create a new DD workspace.

Note

Each DD workspace name consists of the name of the project file and the predefined Workspace Name, which is derived from its Workspace ID. The DD workspace has the next available Workspace Name, for example, DD5 if DD4 was the last DD workspace used. You can add an optional descriptive name via the SetDDAttribute DD MATLAB API function.

3 In the **Select DD Project File Template** dialog, select either a predefined template or another template for the new DD workspace and click **OK**.

Result

You created a DD workspace.

Related topics

Basics

HowTos

References

dsdd_manage_project('Open', projectFile) (📖 TargetLink API Reference)
New - Create New DD Workspace (📖 TargetLink Data Dictionary Manager Reference)
New – Use Active DD Workspace (📖 TargetLink Data Dictionary Manager Reference)
Topic Settings (📖 TargetLink Tool and Utility Reference)

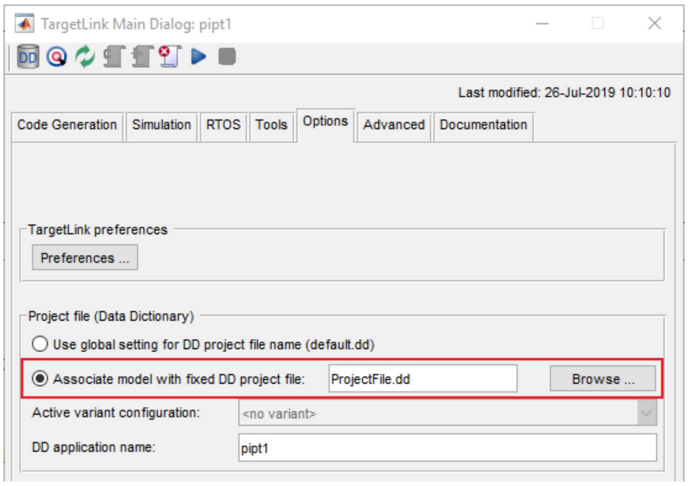
How to Associate a DD Project File with a TargetLink Model

Method

To associate a DD project file with a TargetLink model

1 In the TargetLink Main Dialog, select the Options page.

2 In the Project file (Data Dictionary) frame, click Browse and select a DD project file.


TargetLink Main Dialog: pipt1

Last modified: 26-Jul-2019 10:10:10

Use global setting for DD project file name (default.dd)

Associate model with fixed DD project file: ProjectFile.dd

Active variant configuration: <no variant>

DD application name: pipt1

Result

You associated a DD project file with a TargetLink model.

Related topics

Basics

Basics on the Data Dictionary Manager... 31

How to Open DD Project Files in DD Workspaces

Objective

To open DD project files in DD workspaces.

Method

To open DD project files in DD workspaces

1 From the menu bar of the Data Dictionary Manager, select File – Open.

2 Select one of the following options:

- Click Open - Use Active DD Workspace to clear the active DD workspace and to open a DD project file in it. The DD workspace name, e.g. DD0, does not change.

- Click Open - Create New DD Workspace to open an existing DD project file in a new DD workspace.

  Note

  Each DD workspace name consists of the name of the project file and the predefined Workspace Name, which is derived from its Workspace ID. The DD workspace has the next available Workspace Name, for example, DD5 if DD4 was the last DD workspace used. You can add an optional descriptive name via the SetDDAttribute DD MATLAB API function.

3 In the Open dialog, select the DD project file and click Open.

Result

You opened DD project files in DD workspaces.

Note

Included DD files that are referenced from the opened DD project file are also loaded if the AutoLoad properties of the DD IncludeFile objects are set to on.

Related topics

Basics

HowTos

References

Open - Create New DD Workspace (TargetLink Data Dictionary Manager Reference)
Open - Use Active DD Workspace (TargetLink Data Dictionary Manager Reference)

How to Save DD Workspaces to DD Project Files

Objective

To save DD workspaces to DD project files.

The Data Dictionary Manager provides different ways to save DD workspaces:

- You can save a single DD workspace to a DD project file.
- You can save all open DD workspaces to separate DD project files.
- You can save specific DD objects from the DD object tree to a partial DD file.

Possible methods

Use one of the following ways to save a complete DD project file:

- To save the active DD workspace to a DD project file, refer to Method 1 on page 62.
- To save the active DD workspace using a different name and/or a different folder, refer to Method 2 on page 62.
- To save all open DD workspaces, even existing DD workspaces that are not displayed in a separate pane yet, or that do not have an associated DD project file, refer to Method 3 on page 63.

Tip

You can also perform workspace operations such as Save, Save As, or Save and Close via the DD Workspace Overview pane. Refer to DD Workspace Overview.

Method 1

To save the active DD workspace to a DD project file

1 From the menu bar of the Data Dictionary Manager, select File – Save:
   - If the active DD workspace was already saved, it is saved under the existing name.
   - If it has not been saved yet, the Save As dialog opens for you to enter a new name.

Method 2

To save the active DD workspace using a different name and/or folder

1 From the menu bar of the Data Dictionary Manager, select File – Save As.

2 The Save As dialog opens for you to select a folder and to enter a file name.

3 Click Save to save the active DD workspace and close the dialog.

Note

If an admin password is set, it is also written to the DD project file and restored when you open the file again.

Method 3

To save all open DD workspaces

1 From the menu bar of the Data Dictionary Manager, select File – Save All.

All open DD workspaces are saved with their existing names. If no name is assigned to a specific DD workspace yet, the Save As dialog opens for you to enter a new file name.

Result

You saved DD workspaces to DD project files.

Note

Included DD files are also saved if the AutoSave property is set to On or to OnlyIfModified. If the AutoSave property is set to PromptIfModified you are asked if you want to save your changes.

Related topics

HowTos

References

DD Workspace Overview (TargetLink Data Dictionary Manager Reference)
Save All (TargetLink Data Dictionary Manager Reference)
Save As (TargetLink Data Dictionary Manager Reference)

How to Save a DD Subtree to a Partial DD File

Objective

To save a DD subtree to a partial DD file.

Method

To save a DD subtree to a partial DD file

1 In the DD object tree of the Data Dictionary Manager, select the parent DD object of the DD subtree you want to save to a partial DD file.

2 On the context menu, select Save As.

3 In the Save As dialog, enter the file name or select the file to which you want to save the DD subtree.

4 Click Save to close the dialog.

Result

You saved a DD subtree to a partial DD file.

Next Step

After you saved a DD subtree to a partial DD file, you can open or load this partial DD file to a suitable DD object of another DD workspace. You can use it as an included DD file.

Related topics

Basics

HowTos

References

Point of Inclusion (TargetLink Data Dictionary Manager Reference)
Save As (TargetLink Data Dictionary Manager Reference)

Centrally Manage and Share Partial DD Files

Where to go from here

Information in this section

How to Include Partial Data Dictionary Files

Objective

To load data from an external partial DD file to a specific DD object in a DD workspace each time the DD project file is opened. In this context, a partial DD file is called included DD file and can be located on a central network server where all team members can share the same configuration and pool data.

Protecting included DD files

You can protect the included DD files against modifications by setting the DD access rights (see Basics on User Modes and Access Rights on page 19) or by setting the included DD files to read-only on the file system level on the network server.

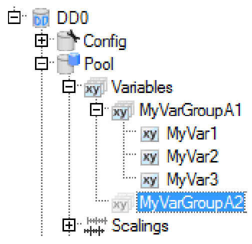
Preconditions

The partial DD file that you want to include in your DD project file must already exist. Refer to How to Save a DD Subtree to a Partial DD File on page 63.

Method

To include partial Data Dictionary files

1 Select the DD object you want to include your partial DD file to.

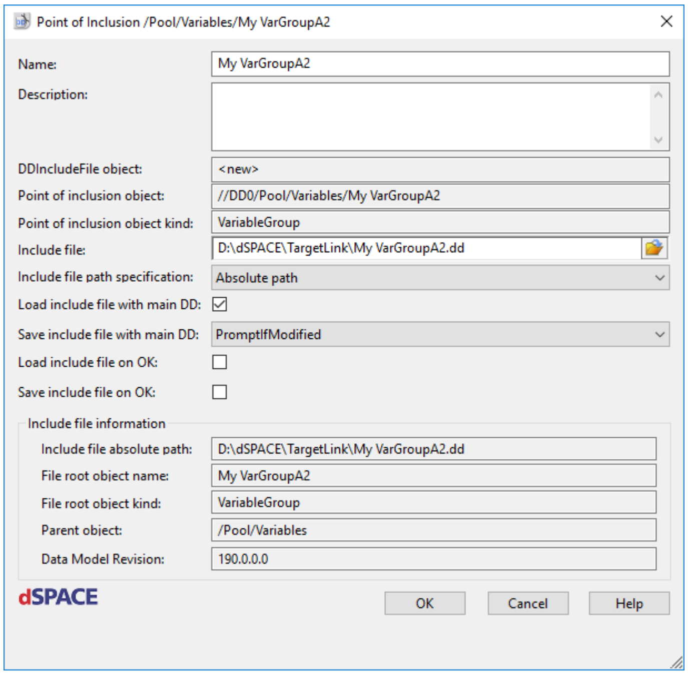


2 From the context menu of the DD object, select Point of Inclusion - Make Point of Inclusion.

Note

The Config, the / Config / DDIncludeFiles, and the Pool objects cannot be made points of inclusion. Files whose file root object is a Config, Pool, or DD root object cannot be included.

The Point of Inclusion dialog opens.


Point of Inclusion /Pool/Variables/My VarGroupA2

Name:	My VarGroupA2
Description:	
DDIncludeFile object:	<new>
Point of inclusion object:	//DD0/Pool/Variables/My VarGroupA2
Point of inclusion object kind:	VariableGroup
Include file:	D:\dSPACE\TargetLink\My VarGroupA2.dd
Include file path specification:	Absolute path
Load include file with main DD:	☑
Save include file with main DD:	PromptIfModified
Load include file on OK:	☐
Save include file on OK:	☐

Include file information

Include file absolute path:	D:\dSPACE\TargetLink\My VarGroupA2.dd
File root object name:	My VarGroupA2
File root object kind:	VariableGroup
Parent object:	/Pool/Variables
Data Model Revision:	190.0.0.0

3 In the Point of Inclusion dialog, choose a partial DD file you want to include by clicking the Browse button.

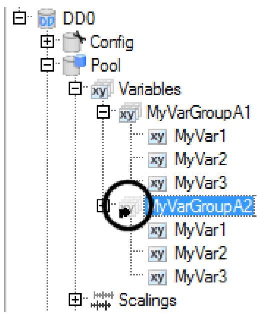
4 Select the Include file path specification if you want to change the way the file path is specified. This is helpful if the partial DD file is located on a network server. The default setting is that the path is specified relative to the main DD file path.

5 Select the Load include file with main DD checkbox if you want to automatically load the included DD file when the main DD file is opened the next time (autoload).

6 Select the Load Include File on OK checkbox to immediately load the partial DD file into your current DD workspace.

7 Click OK to include the partial DD file into your current DD workspace.

As shown in the illustration below, the small arrow indicates an inclusion point.



Tip

Managing multiple partial DD files

- If you want to include multiple partial DD files, you can use the Open command and multiselect all partial DD files to load them into your current DD workspace. You can then use the Point of Inclusion dialog to include each of the loaded partial DD files. Because of the metadata saved within the partial DD files, the mapping of the files is set automatically.
- Another method to manage points of inclusion is to modify their corresponding DDIncludeFile objects in the `Config/DDIncludeFiles` object tree. For more information, refer to DDIncludeFile Object Dialog (📖 TargetLink Data Dictionary Manager Reference).

Result

The specified partial DD file is included in the DD object. The next time you open the DD project file, the included data is automatically available.

For more information on the functions and properties of the Point of Inclusion dialog, refer to Point of Inclusion (📖 TargetLink Data Dictionary Manager Reference).

Note

If one or more partial DD files could not be included in the DD objects, a message is displayed in the Message Browser. It displays more information on the affected file.

Related topics

Basics

HowTos

References

Point of Inclusion (📖 TargetLink Data Dictionary Manager Reference)

How to Separate DD Subtrees and Save Them as Included DD Files

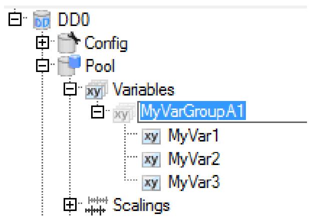
Objective

You can separate specific DD subtrees from your current DD workspace and save them as an included DD file.

Method

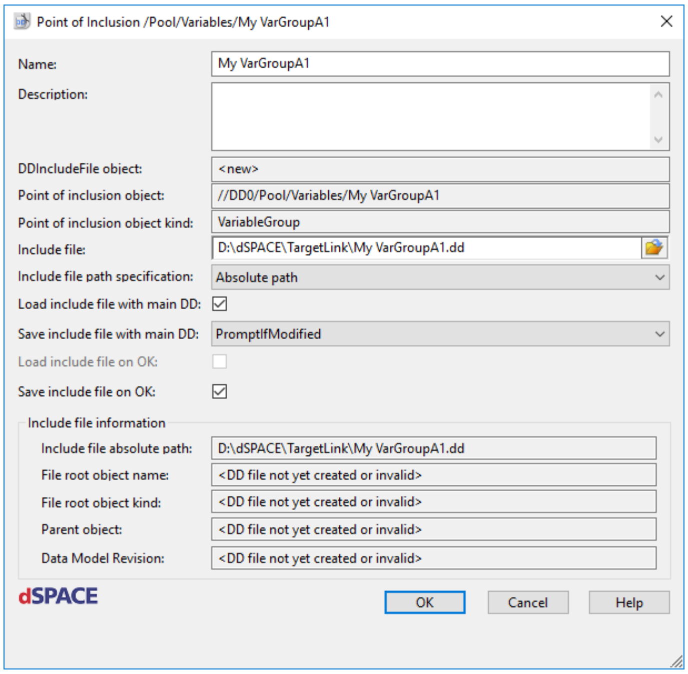
To separate DD subtrees and save them as included DD files

1 Select the DD subtree you want to separate.



2 From the context menu of the DD object, select Point of Inclusion - Make Point of Inclusion.

The Point of Inclusion dialog opens.



Point of Inclusion /Pool/Variables/My VarGroupA1

Name:	My VarGroupA1
Description:	
DDIncludeFile object:	<new>
Point of inclusion object:	//DD0/Pool/Variables/My VarGroupA1
Point of inclusion object kind:	VariableGroup
Include file:	D:\dSPACE\TargetLink\My VarGroupA1.dd
Include file path specification:	Absolute path
Load include file with main DD:	✓
Save include file with main DD:	PromptIfModified
Load include file on OK:	
Save include file on OK:	✓

Include file information

Include file absolute path:	D:\dSPACE\TargetLink\My VarGroupA1.dd
File root object name:	<DD file not yet created or invalid>
File root object kind:	<DD file not yet created or invalid>
Parent object:	<DD file not yet created or invalid>
Data Model Revision:	<DD file not yet created or invalid>

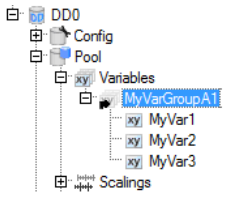
3 Select the Load include file with main DD checkbox if you want to automatically load the included DD file when the main DD file is opened the next time (autoload).

4 In the Include file field, enter a name for the included DD file you want to create. By default, the name of the selected DD object is used.

5 Select the Save Include File on OK checkbox to separate the DD subtree immediately after clicking OK.

6 Click OK to separate the DD object tree and make it an included DD file.

As shown in the illustration below, the small arrow indicates an inclusion point.



Result

You have separated a specific DD subtree from your current DD workspace and saved it as an included DD file.

Related topics

Basics

Basics on Opening and Handling DD Files.......... 54

HowTos

How to Include Partial Data Dictionary Files.......... 65

References

Point of Inclusion (📖 TargetLink Data Dictionary Manager Reference)

How to Group DDIncludeFiles Objects

Objective

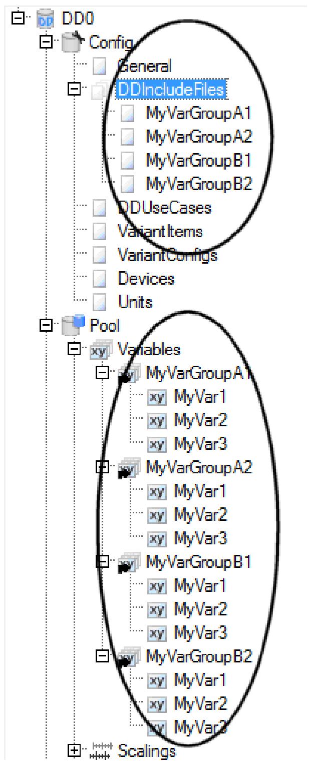
You can group **DDIncludeFile** objects any way you like. **DDIncludeFile** objects are used to specify inclusion points. They contain data about the included DD file path and the DD object tree where the inclusion point is located. Grouping **DDIncludeFile** objects help you structure these objects and makes it easier to transfer inclusion points from one DD workspace to another.

Method

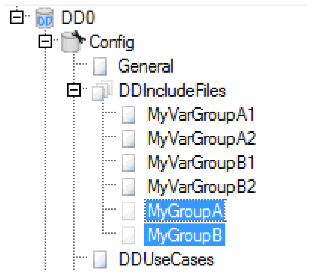
To group DDIncludeFile objects

1 In your DD workspace, navigate to `Config/DDIncludeFiles`.

As shown in the example below, you can see four DDIncludeFile objects and their corresponding inclusion points in the `Variables` subtree.

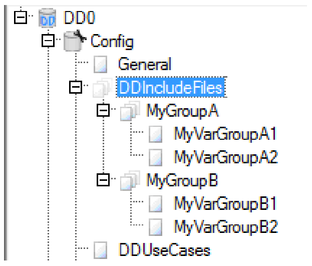


2 From the context menu of the DDIncludeFile object, select Create DDIncludeFileGroup.



3 Name the new object and repeat steps 2 and 3 if necessary.

4 Drag and drop the DDIncludeFile objects into the DDIncludeFileGroup objects you created.



Result

You have grouped DDIncludeFile objects.

Managing Data Dictionary Objects

Introduction

The TargetLink Data Dictionary Manager helps you to handle DD objects by actions such as creating, editing, copying, and renaming them. The instructions given below might not work for all the DD objects defined in the data model. The actions which are allowed for any specific DD object depend on its kind and its position in the DD object tree.

Where to go from here

Information in this section

How to Create New Data Dictionary Objects

Introduction

You can use the **TargetLink Data Dictionary Manager** to create new DD objects in the DD object tree.

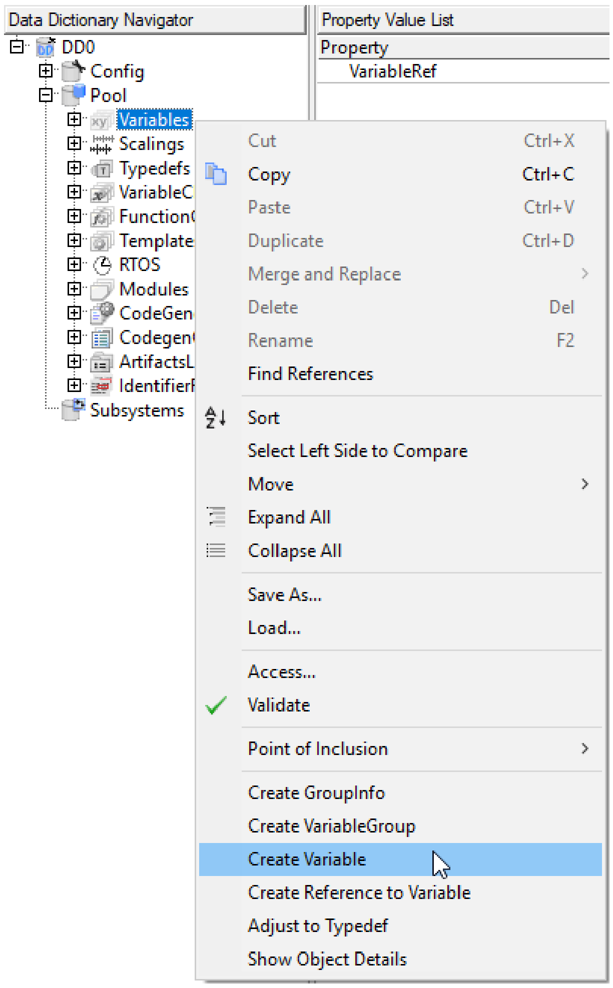
Creating new DD objects

New DD objects can be created for all DD objects except for the root DD object. The kind and number of objects which can be created depend on the object selected in the DD object tree. For the `/Pool/Variables` variable group, you can create the following objects: **GroupInfo**, **Variable**, **VariableGroup**. In the following instructions, <DDObject> represents an object which can be created. In addition, you can also create Variable objects in block dialogs. Refer to How to Create DD Variable Objects via Block Dialog (📖 TargetLink Preparation and Simulation Guide).

Method

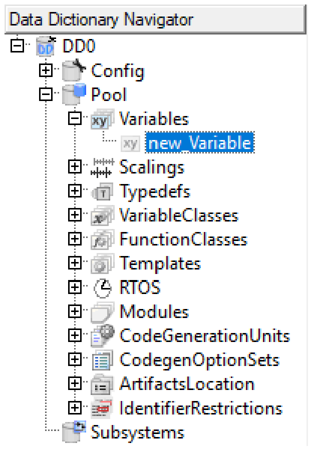
To create a new DD object

1. In the DD object tree, select the DD parent object you want to add a new DD object to.
2. From the context menu of the selected DD object, select the **Create <DDObject>** command you want to use.



Result

The selected DD object is created in the **TargetLink Data Dictionary**.


Data Dictionary Navigator
- DD0
  - Config
  - Pool
    - Variables
      - new_Variable
    - Scalings
    - Typedefs
    - VariableClasses
    - FunctionClasses
    - Templates
    - RTOS
    - Modules
    - CodeGenerationUnits
    - CodegenOptionSets
    - ArtifactsLocation
    - IdentifierRestrictions
  - Subsystems

Next step

In the DD object tree, the new DD object is displayed with a default name below your selected DD parent object. This name for this example consists of the prefix `new_` and the object type you just added. You can edit the default name if the data model allows this. For more information, refer to How to Rename Data Dictionary Objects on page 77.

Related topics

HowTos

How to Create DD Variable Objects via Block Dialog (TargetLink Preparation and Simulation Guide)
How to Rename Data Dictionary Objects.. 77

References

Create <DD Object> (TargetLink Data Dictionary Manager Reference)

How to Set Properties of Data Dictionary Objects

Objective

The properties of DD objects can be set individually.

Possible methods

You can choose between two methods when setting properties for DD objects.

- You can configure DD objects by means of dialog settings which guide you through configuration. For more information, refer to Method 1.

- The fastest way to set the properties of DD objects is via the Property Value List. For more information, refer to Method 2.

Method 1

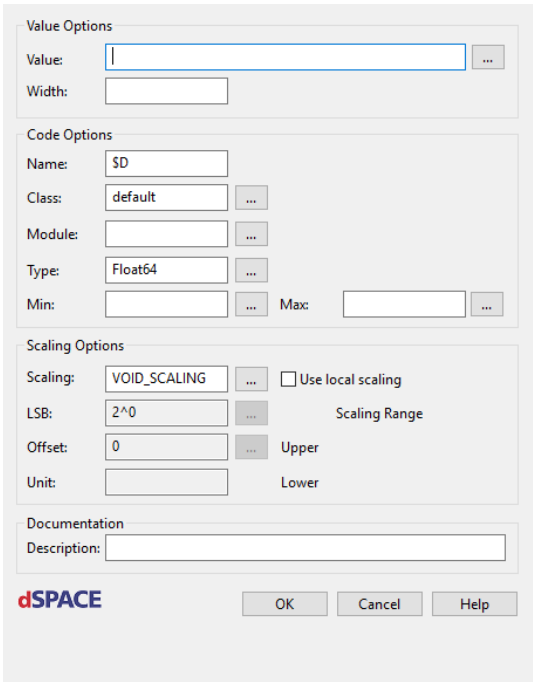
To set properties of a DD object via a dialog

Note

Dialogs to set properties are not available for all DD objects. For information on which DD objects can be defined via dialogs, refer to Basic Dialogs (TargetLink Data Dictionary Manager Reference).

1 In the DD object tree, double-click the DD object whose properties you want to edit.

The dialog for the selected DD object opens, for example, a Plain Variable dialog.


Value Options
Value:
Width:
Code Options
Name: $D
Class: default
Module:
Type: Float64
Min:
Max:
Scaling Options
Scaling: VOID_SCALING
Use local scaling
LSB: 2^0
Scaling Range
Offset: 0
Upper
Unit:
Lower
Documentation
Description:

2 In the dialog, set the properties.

Method 2

To set properties of a DD object via the Property Value List

1 In the DD object tree, select the object whose properties you want to edit.

2 In the Property Value List, edit the properties' value by selecting a value from a drop-down list, by clicking the edit button, or by selecting the Open Editor command in the context menu.

3 To validate the changes you made click the Validate ✓ icon in the toolbar.

Result

The properties are set according to your entries.

Related topics

References

Basic Dialogs (TargetLink Data Dictionary Manager Reference)
Open Editor (TargetLink Data Dictionary Manager Reference)
Plain Variable (TargetLink Data Dictionary Manager Reference)
Validate (TargetLink Data Dictionary Manager Reference)

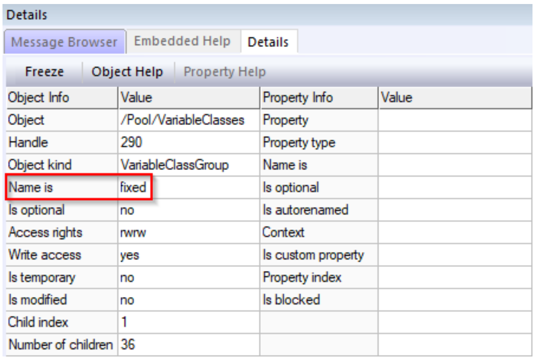
How to Rename Data Dictionary Objects

Objective

You can rename existing DD objects in the DD object tree, if the data model allows this.

Renaming allowed

You can check whether a DD object can be renamed in the **Details** (TargetLink Data Dictionary Manager Reference) pane. The **Name is:** field indicates, whether you can rename the object (arbitrary) or not (fixed). In the example below, you cannot rename the object.



Object Info	Value	Property Info	Value
Object	/Pool/VariableClasses	Property	
Handle	290	Property type	
Object kind	VariableClassGroup	Name is	
Name is	fixed	Is optional	
Is optional	no	Is autorenamed	
Access rights	rwrw	Context	
Write access	yes	Is custom property	
Is temporary	no	Property index	
Is modified	no	Is blocked	
Child index	1		
Number of children	36		

Valid references

If you rename a DD object, its reference via a DD path normally becomes invalid. However, you can stop references from becoming invalid if you use the **Reference Handling Options** to specify rules for adapting them. For example, you can let the Data Dictionary Manager automatically adapt all the references. For details, refer to Reference Handling Options (TargetLink Data Dictionary Manager Reference).

Method

To rename a DD object

1 Select the DD object you want to rename.

2 From the menu bar, select Edit – Rename or press `F2`.

3 Enter a new name.

Result

The selected DD object is displayed with the new name.

Related topics

References

Details (📖 TargetLink Data Dictionary Manager Reference)
Reference Handling Options (📖 TargetLink Data Dictionary Manager Reference)
Rename (📖 TargetLink Data Dictionary Manager Reference)

How to Delete Data Dictionary Objects

Introduction

DD objects can be deleted when they are no longer of use and if the data model allows this.

Note

- If a DD object that is referenced is deleted, the reference becomes invalid.
- If you delete a parent DD object, all its children are also automatically deleted.
- A deleted DD object cannot be restored. There is no undo function.

Method

To delete a DD object

1 Select the DD object you want to delete.

2 From the menu bar, select Edit – Delete or press `Del`.

3 Click OK to confirm the deletion.

Result

The selected DD object is deleted.

Related topics

References

Delete (📖 TargetLink Data Dictionary Manager Reference)

How to Find Data Dictionary Objects

Introduction

If you know the name of a DD object, the name of a property, or a property value, you can look for it in the DD object tree.

Regular expressions and wildcards

If you know only a part of the object name, property name or property value, you can use regular expressions or wildcards in your search strings.

Examples of regular expression The search string must comply with the regular expression pattern.

`[A-Z]*`	To search for objects whose names consist of capital letters.
`MyObj.*`	To search for objects whose names start with *MyObj*.
`.*MyObj.*`	To search for objects whose names contain the string *MyObj*.
`M..bj`	To search for objects whose names start with *M*, continue with any two characters and end with *bj*.

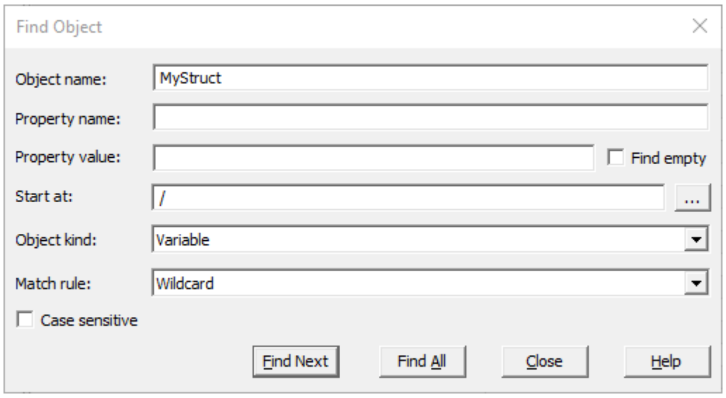
Examples of wildcards In the search string, ? matches any single character whereas * matches 0, 1 or any number of characters.

`MyObj*`	To search for objects whose name starts with *MyObj*.
`*MyObj*`	To search for objects whose name contains the string *MyObj*.
`M??bj`	To search for objects whose names start with *M*, continue with any two characters and end with *bj*.

Method

To find a DD object

1 From the menu bar, select Edit –Find or press **`Ctrl`** + **`F`**.


Find Object

Object name: MyStruct
Property name:
Property value: ☐ Find empty
Start at: /
Object kind: Variable
Match rule: Wildcard
☐ Case sensitive

2 Enter the Object name or Property name and/or the Property value you are looking for. Use regular expressions or wildcards if necessary. If you are looking for a property that is empty, you have to select the Find empty checkbox, because empty inputs are ignored during the search.

Note

All input fields that are not empty (except Find empty) must apply in the search so that the object is identified as found. Therefore, all non-empty inputs are linked by Boolean AND.

3 Click the Browse button to define where the search should be started.
4 Select the Object kind of the DD object you are looking for.
5 From the Match rule list, select the rule to be applied.
6 Select the Case sensitive checkbox that are useful for your search.
7 Click Find Next or press **F3** to start the search.
The Data Dictionary Manager searches the DD object in the data dictionary. If a DD object that fulfills *all* the search criteria is found, it is selected in the DD object tree.

Result

The DD object you are looking for is selected in the DD object tree.

Tip

If you want to find all the objects that match the specified search criteria, you can click Find All. The **Find Object Results** (TargetLink Data Dictionary Manager Reference) pane opens and displays the search results.

Related topics

References

Find (TargetLink Data Dictionary Manager Reference)
Find Next (TargetLink Data Dictionary Manager Reference)

How to Validate DD Objects

Objective

The TargetLink Data Dictionary enables you to check if your DD objects conform to the data model. Validation checks if the DD contains valid data.

Method

To validate DD objects

1 In the DD object tree, select the DD object to be validated.
2 In the context menu, select Validate or click the ✓ icon in the Data Dictionary Manager toolbar to start validation for the selected DD object.

Result

The DD object tree is validated.

Note

Invalid DD objects or dependencies are not corrected or deleted during validation. If the DD object tree is not validated successfully, all messages are displayed in the Message Browser

If you want to fix the invalid objects automatically, you can use the DD MATLAB API Function **`Validate`** with the **`fix`** option set to on as shown below.

```
[bSuccess,vs] = dsdd('Validate','/poscontrol','fix','on','Validat
ionLevel',4);
>> bSuccess,vs
bSuccess = 1
vs = numOfValidatedObjects: 1
   numOfNotValidatedObjects: 0
   numOfInvalidObjects: 0
   numOfFixedObjects: 0
```

Note that not all defects can be fixed automatically.

Related topics

Basics

Basics on Validating Data Dictionaries.. 26

References

Validate (TargetLink Data Dictionary Manager Reference)

How to Reference Data Dictionary Objects

Objective

There are several references between DD objects. For example, a Variable object can reference a Typedef or a Scaling object.

Specifying references via DD path and DD handle

References can be specified by either a DD path or a DD handle (unsigned 32-bit number). For information on the DD handle and the DD path, refer to Basics on Accessing DD Objects via Paths and Handles on page 24.

Method

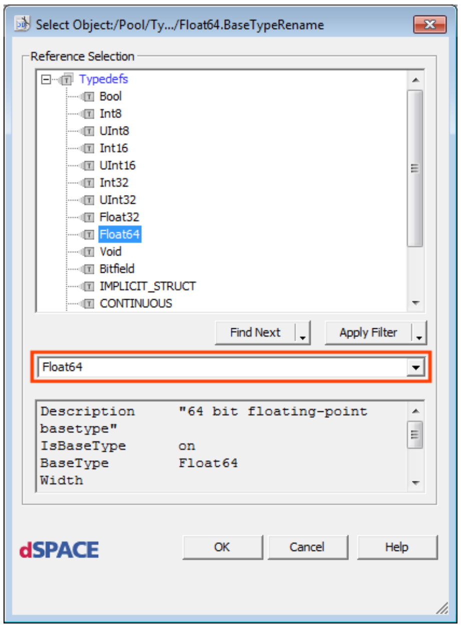
To reference a DD object

1 In the DD object tree, select the DD object you want to create a reference for, for example, `/Pool/Variables/MyVar`.

2 In the **Property Value List**, select the property you want to set, for example, **Type**.

3 Click the **Edit** button at the right side of the property value edit field.

The **Select Object** dialog opens and displays all the objects you can set at the selected object.



```
Description     "64 bit floating-point
basetype"
IsBaseType      on
BaseType        Float64
Width
```

Tip

You can filter and/or search the displayed DD reference objects via the **Filter** edit field and the **Find Next** and **Apply Filter** buttons.

4 Select an object and click **OK** to close the dialog.

Result

The properties of the selected reference are set. The reference is stored as a path.

Note

If you move or rename a DD object that is referenced via a DD path, the reference normally becomes invalid. However, the Data Dictionary Manager automatically adapts all the references to keep them valid. You can change this behavior via the **Reference Handling Options** to specify rules for adapting them. For details, refer to Reference Handling Options (📖 TargetLink Data Dictionary Manager Reference).

Tip

If you want to set a reference by handle, you can type the DD handle in the edit field of the Property Value List of the Data Dictionary Manager. Keep in mind that DD handles must not be used in the `/Pool` and `/Config` area.

Related topics

Basics

HowTos

References

How to Find Data Dictionary Object References

Introduction

If you want to know which other objects a DD object is referenced by, you can search for references.

Preconditions

To perform the steps described below, the following preconditions must be fulfilled:

- The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.
- Objects in a model, such as Simulink blocks or Stateflow objects, that reference a DD object can be found only if the Data Dictionary Manager was started via MATLAB and the model is connected to the current DD.

Method

To find a DD object reference

1. In the DD Navigator, right-click the DD object whose references you want to find.
2. From the context menu, select Find References.

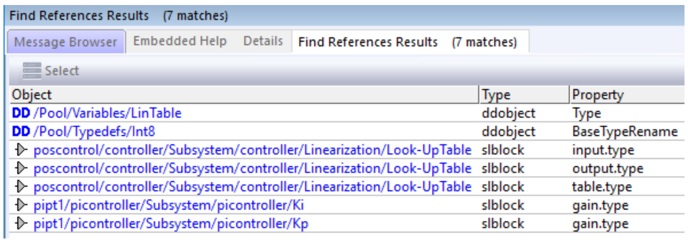
Result

The Find References Results pane opens and displays the references to the selected object.

Examples

In the following example, there are 5 references to the Int8 Typedefs object:

- Other DD objects
- Simulink blocks that are based on the selected Typedefs object



Find References Results (7 matches)

Message Browser | Embedded Help | Details | Find References Results (7 matches)

Select

Object	Type	Property
DD /Pool/Variables/LinTable	ddobject	Type
DD /Pool/Typedefs/Int8	ddobject	BaseTypeRename
poscontrol/controller/Subsystem/controller/Linearization/Look-UpTable	slblock	input.type
poscontrol/controller/Subsystem/controller/Linearization/Look-UpTable	slblock	output.type
poscontrol/controller/Subsystem/controller/Linearization/Look-UpTable	slblock	table.type
pipt1/picontroller/Subsystem/picontroller/Ki	slblock	gain.type
pipt1/picontroller/Subsystem/picontroller/Kp	slblock	gain.type

Related topics

HowTos

How to Reference Data Dictionary Objects.. 81

References

Find References (TargetLink Data Dictionary Manager Reference)
Find References Results (TargetLink Data Dictionary Manager Reference)
tlFindDDReferences (TargetLink API Reference)

Example of Creating New Typedefs

Introduction

To define a new Typedef object.

Method

To create a new Typedef object

1 In the DD object tree, select the `/Pool/Typedefs` object.

2 From the context menu, select Create Typedef.
A new Typedef object is created in the DD object tree.

3 Enter a name for the new Typedef object, for example **`MyStructType`**.

4 Double-click the new Typedef object.

5 In the Plain Typedef dialog, select the base type Struct.

6 In the Struct Typedef dialog, add the struct components.

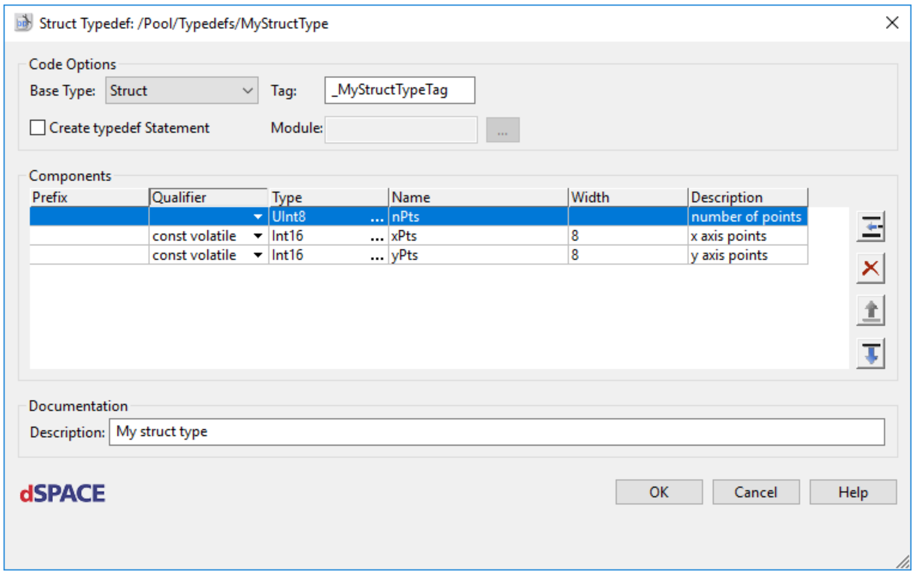
7 Enter the names and properties of the struct components.

Note

The width for bitfield components must be 1.

8 Select the Create typedef statement checkbox, if required.

9 Enter a descriptive text , for example, `My struct type` as description.


Struct Typedef: /Pool/Typedefs/MyStructType

Code Options

Base Type: Struct   Tag: _MyStructTypeTag

☐ Create typedef Statement   Module:

Components

Prefix	Qualifier	Type	Name	Width	Description
		UInt8	nPts		number of points
	const volatile	Int16	xPts	8	x axis points
	const volatile	Int16	yPts	8	y axis points

Documentation

Description: My struct type

10 Click OK.

Result

The new Typedef object with the specified components.

Related topics

References

Create <DD Object> (📖 TargetLink Data Dictionary Manager Reference)
Plain Typedef (📖 TargetLink Data Dictionary Manager Reference)
Struct Typedef (📖 TargetLink Data Dictionary Manager Reference)

Example of Creating New Variables

Introduction

To create a new struct Variable object.

Preconditions

A Typedef object of the Struct base type must have been created.

Method

To create a new variable

1 In the DD object tree, select the `/Pool/Variables` object.

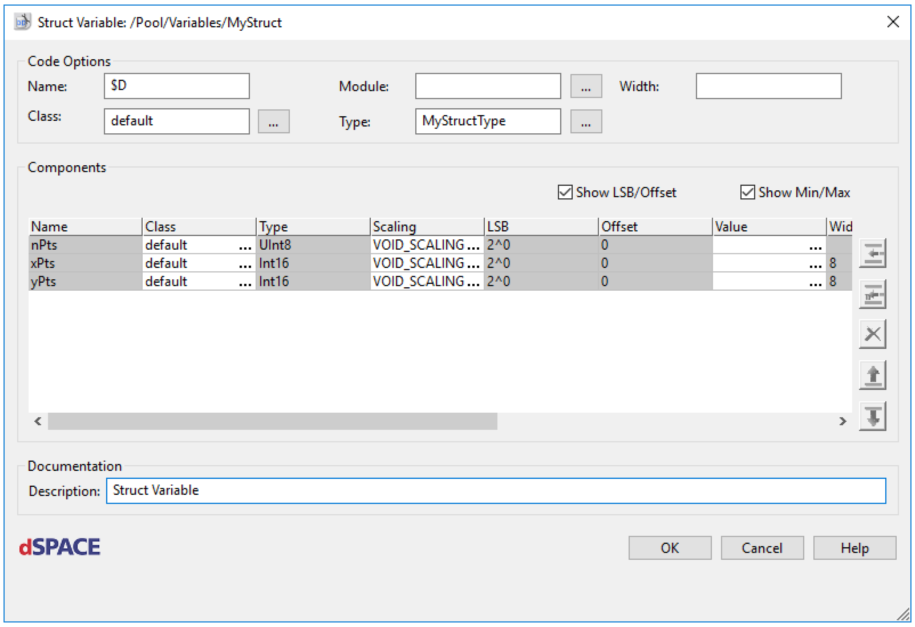
2 In the context menu, select Create Variable.
  A new Variable object is created in the DD object tree.

3 Type the name of the new variable, for example, **`MyStruct`**.

4 Double-click the new Variable object.

5 In the Plain Variable dialog, select an existing struct type, for example, MyStructType.

6 In the Struct Variable dialog, type **`Struct Variable`** as the description.



Struct Variable: /Pool/Variables/MyStruct

Code Options

Name:	$D	Module:		Width:	
Class:	default	Type:	MyStructType		

Components

☑ Show LSB/Offset ☑ Show Min/Max

Name	Class	Type	Scaling	LSB	Offset	Value	Wid
nPts	default ...	UInt8	VOID_SCALING ...	2^0	0	...	
xPts	default ...	Int16	VOID_SCALING ...	2^0	0	...	8
yPts	default ...	Int16	VOID_SCALING ...	2^0	0	...	8

Documentation

Description: Struct Variable

7 Click OK.

Result

The new struct Variable object is created.

Related topics

References

Create <DD Object> (📖 TargetLink Data Dictionary Manager Reference)
Plain Variable (📖 TargetLink Data Dictionary Manager Reference)
Struct or Implicit Struct Variable (📖 TargetLink Data Dictionary Manager Reference)

Comparing and Merging Data Dictionaries

Where to go from here

Information in this section

Basics on Comparing and Merging DD Objects and Workspaces Using the DD Two-Way Comparison

Comparing and merging using the DD two-way comparison

To compare and merge DD objects or DD workspaces, you can use the DD two-way comparison. It displays a synchronized tree structure of the compared DD workspaces, including their DD child objects with properties. By using the DD two-way comparison, you can do the following:

- Compare and merge selected DD objects in one DD workspace.
- Compare and merge selected DD objects in different DD workspaces.
- Compare and merge entire DD workspaces.

Starting the DD two-way comparison

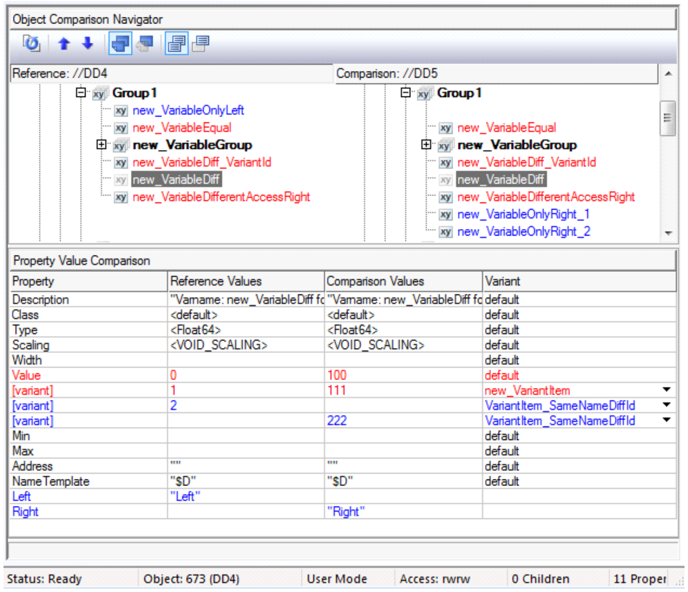
You can start the DD two-way comparison as follows:

Method	Instructions
DD Manager	Refer to How to Compare DD Objects Using the DD Two-Way Comparison on page 93.

Method	Instructions
API	Refer to dsddman('Compare', propertyName, propertyValue, ...).
Windows Command Prompt	Change the directory to `<InstallationDir>\dsdd\bin` and use the `DsddMan.exe` command. To compare two DD files, enter `DsddMan <file1> <file2>`.[1]
Version control system (VCS)	Refer to Basics on Configuring Version Control Systems to Compare and Merge DD Files on page 90.

[1] Enter `DsddMan /?` to display a help window with a list of all possible command line arguments.

Working with the DD two-way comparison



Object Comparison Navigator

Reference: //DD4	Comparison: //DD5
Group 1	Group 1
new_VariableOnlyLeft	
new_VariableEqual	new_VariableEqual
new_VariableGroup	new_VariableGroup
new_VariableDiff_VariantId	new_VariableDiff_VariantId
new_VariableDiff	new_VariableDiff
new_VariableDifferentAccessRight	new_VariableDifferentAccessRight
	new_VariableOnlyRight_1
	new_VariableOnlyRight_2

Property Value Comparison

Property	Reference Values	Comparison Values	Variant
Description	"Varname: new_VariableDiff fc	"Varname: new_VariableDiff fc	default
Class	<default>	<default>	default
Type	<Float64>	<Float64>	default
Scaling	<VOID_SCALING>	<VOID_SCALING>	default
Width			default
Value	0	100	default
[variant]	1	111	new_VariantItem
[variant]	2		VariantItem_SameNameDiffId
[variant]		222	VariantItem_SameNameDiffId
Min			default
Max			default
Address	""	""	default
NameTemplate	"$D"	"$D"	default
Left	"Left"		
Right		"Right"	

Status: Ready | Object: 673 (DD4) | User Mode | Access: rwrw | 0 Children | 11 Proper

The DD two-way comparison consists of the Object Comparison Navigator and the Property Value Comparison table. The Object Comparison Navigator contains a Reference: <root path> column on the left-hand side and a Comparison: <root path> column on the right-hand side. Differences of the compared DD objects are indicated by differently colored text.

Tip

Use filter rule sets to do the following:

- To hide DD objects and properties of no interest.
- To easily compare a large number of DD objects.

Refer to Basics on Filter Rule Sets for the Data Model on page 121.

Color coding The following colors indicate differences between the compared DD objects, properties, and attributes:

Color	Description
Red	A corresponding DD object or value exists, but the DD objects or values differ.
Blue	No corresponding DD object or value exists.
Black, bold	The corresponding DD object is identical, but some descendants differ.
Red, bold	The corresponding DD object and descendants differ.

Customizing the comparison

Use the Data Dictionary Compare Tool Options dialog to customize your comparison. Refer to Compare Tool Options (📖 TargetLink Data Dictionary Manager Reference).

Related topics

Basics

Basics on Filter Rule Sets for the Data Model.. 121

HowTos

How to Generate a Comparison Report Using the DD Two-Way Comparison............................. 97

References

Compare with <DD Object> (📖 TargetLink Data Dictionary Manager Reference)
dsddman('Compare', propertyName, propertyValue, ...) (📖 TargetLink API Reference)

Basics on Configuring Version Control Systems to Compare and Merge DD Files

Introduction

If you want to perform a difference analysis or a three-way merge of DD files with a version control system, you have to configure it.

> **Note**
>
> To compare and merge two different DD files without a version control system, you can do the following:
> - Use the DD two-way comparison. Refer to Basics on Comparing and Merging DD Objects and Workspaces Using the DD Two-Way Comparison on page 87.
> - Use the DD three-way merge, if you also have a common ancestor DD file. Refer to Basics on Comparing and Merging DD Files via DD Three-Way Merge on page 98.

Configuration of common version control systems

The location of the relevant configuration settings of certain version control systems is described in the following table. Refer to the user documentation of your specific version control system.

> **Note**
>
> `<Path to Dsddman.exe>` is the following path: `<Installation Directory of TargetLink>\dsdd\bin\dsddman.exe`.

Version Control System	Configuration for Difference Analysis	Configuration for Three-Way Merge

PTC Integrity	Edit the `IntegrityClientSite.rc` file in your PTC installation folder.
	`diffTools=ddcompare`
	`diffTools.ddcompare.title=DD Comparison Tool, dSPACE GmbH`
	`diffTools.ddcompare.extension=dd`
	`diffTools.ddcompare.commandLine=<Path to Dsddman.exe> "{3}" "{4}" -reftitle "{1}" -comptitle "{2}"`
		Edit the `IntegrityClientSite.rc` file in your PTC installation folder.
		`mergeTools=ddcompare`
		`mergeTools.ddcompare.title=DD 3-way-merge Tool, dSPACE GmbH`
		`mergeTools.ddcompare.extension=dd`
		`mergeTools.ddcompare.commandLine=<Path to Dsddman.exe> -anctitle "{0}" -reftitle "{2}" -comptitle "{1}" -mergedtitle "{3}" -ancfile "{4}" -reffile "{6}" -compfile "{5}" -mergedfile "{8}" -automerge`

Version Control System	**Configuration for Difference Analysis**	**Configuration for Three-Way Merge**

Git	Use the **`git config`** command.
	`git config --global diff.tool ddcompare`
	
	`git config --global difftool.ddcompare.cmd`
	`"\"<Path To DsddMan.exe>"\ \"%CD%`
	`\\$LOCAL\" \"%CD%\\$REMOTE\""`
		Use the **`git config`** command.
		`git config --global merge.tool ddcompare`
		
		`git config --global mergetool.ddcompare.cmd"\"<Path`
		`To DsddMan.exe>\" \"%CD%\`
		`\$LOCAL\" \"%CD%\\$REMOTE\" -ancfile \"%CD%\\$BASE\"`
		`-mergedfile \"%CD%\\$MERGED\"`
		` -automerge"`

Azure DevOps	Refer to the relevant configuration settings dialog.
	Specify the difference tool: Specify `<Path To DsddMan.exe>`.
	Enter the tool arguments, e.g.:
	`"%1" "%2" -reftitle "%6" -comptitle "%7"`
		Refer to the relevant configuration settings dialog.
		Specify the merge tool: Specify `<Path To DsddMan.exe>`.
		Enter the tool arguments, e.g.:
		`"%1" "%2" -ancfile "%3" -mergedfile "%4" -reftitle`
		`"%6" -comptitle "%7" -mergedtitle "%9"`

TortoiseSVN	Refer to the relevant configuration settings dialog.
	Command to be entered, e.g.:
	`"<Path To DsddMan.exe>" "%base" "%mine" -`
	`reftitle %bname -comptitle %yname`
		Refer to the relevant configuration settings dialog.
		Command to be entered, e.g.:
		`"<Path To DsddMan.exe>" "%mine" "%theirs" -ancfile`
		`"%base" -mergedfile "%merged" -reftitle %yname -`
		`comptitle %tname -mergedtitle %mname`

Note

For further questions, or if your version control system is not described in the table above, contact dSPACE Support.

Related topics

How to Merge or Replace DD Objects

Objective

To merge or replace DD objects. The DD objects can reside in the same or different DD workspaces.

Improved comparing and merging of DD objects

For a difference analysis with improved compare and merge options, you can use the DD two-way comparison. Refer to How to Merge or Replace a DD Object Using the DD Two-Way Comparison on page 95.

Method

To merge or replace DD child objects

1 In the Data Dictionary Navigator, right-click the DD object A, whose data you want to transfer, and select Copy.

2 If necessary, select another DD workspace.

3 Right-click the receiving DD object B in the Data Dictionary Navigator and select the appropriate merge option from the Merge and Replace command:

Merge Option	Description
Merge <Object_path> into this object	DD object B receives all data that exists only in DD object A.
Merge <Object_path> into this object, and overwrite properties	DD object B receives all data that exists in DD object A. Property values are overwritten.
Merge <Object_path> into this object, and overwrite objects	DD object B receives all data that exists in DD object A. Property values and DD child objects existing in both B and A are overwritten. However, data only in DD object B is retained.
Replace with <Object_path>	The target DD object B is removed and replaced by the source DD object A.

Note

With this method, you can also use merging to copy property values of a DD object to another DD object.

Result

You merged or replaced DD objects.

Related topics

Basics

Basics on the Data Dictionary Manager.. 31

References

Merge and Replace (TargetLink Data Dictionary Manager Reference)

How to Compare DD Objects Using the DD Two-Way Comparison

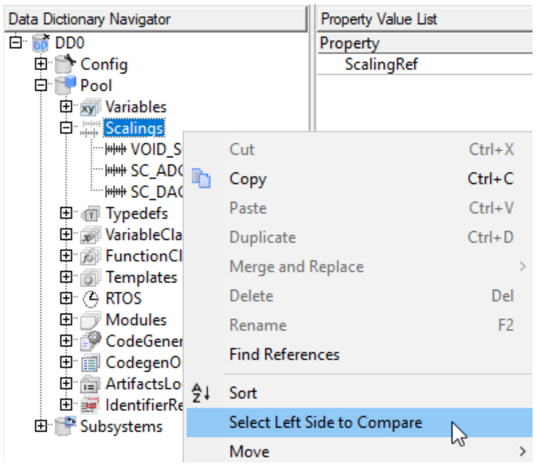
Objective

To compare DD objects, you can use the DD two-way comparison.

Method

To compare DD objects using the DD two-way comparison

1 In the Data Dictionary Navigator, right-click the DD object you want to compare and click Select Left Side to Compare.



2 Select a second DD object for the comparison. It can reside in the same or a different DD workspace.

3 Right-click the selected DD object and click Compare with <DD object>. The DD two-way comparison pane opens.

4 Navigate through the comparison result using the Next/Previous Difference buttons or by selecting the DD objects manually. Corresponding DD objects and the child objects are displayed on the same level and line. Predefined color coding is used to highlight differences between the DD objects.

Refer to Basics on Comparing and Merging DD Objects and Workspaces Using the DD Two-Way Comparison on page 87.

5 Click the Show Objects with differences or Show Properties with Differences button to filter the DD two-way comparison. Only DD objects or properties with differences are displayed. You can clear the filter by clicking the Show All Objects or Show All Properties button (default setting).

Result

You compared DD objects using the DD two-way comparison.

Next Steps

- You can merge DD objects using the DD two-way comparison. Refer to How to Merge or Replace a DD Object Using the DD Two-Way Comparison on page 95.
- You can generate an HTML or XML comparison report. Refer to How to Generate a Comparison Report Using the DD Two-Way Comparison on page 97.

Related topics

Basics

Basics on Comparing and Merging DD Objects and Workspaces Using the DD Two-Way Comparison........ 87

HowTos

How to Generate a Comparison Report Using the DD Two-Way Comparison........ 97
How to Merge or Replace a DD Object Using the DD Two-Way Comparison........ 95

References

Compare with <DD Object> (📖 TargetLink Data Dictionary Manager Reference)
Delete (📖 TargetLink Data Dictionary Manager Reference)
dsddman('Compare', propertyName, propertyValue, ...) (📖 TargetLink API Reference)
Generate Report (HTML/XML) (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle> Without Overwrite (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle>, and Overwrite Objects (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle>, and Overwrite Properties (📖 TargetLink Data Dictionary Manager Reference)
Replace <Left/Right/Middle> (📖 TargetLink Data Dictionary Manager Reference)
Select Left Side to Compare (📖 TargetLink Data Dictionary Manager Reference)
Show in DD Workspace Pane (📖 TargetLink Data Dictionary Manager Reference)

How to Merge or Replace a DD Object Using the DD Two-Way Comparison

Objective

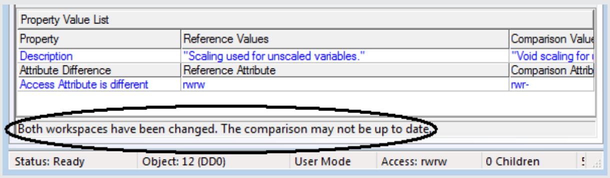
To merge or replace a DD object, you can use the DD two-way comparison.

Precondition

You compared DD objects in the DD two-way comparison. Refer to How to Compare DD Objects Using the DD Two-Way Comparison on page 93.

Note

If you change the contents of DD objects in DD workspaces that are part of the current comparison, the comparison does not refresh automatically:



Property Value List

Property	Reference Values	Comparison Valu
Description	"Scaling used for unscaled variables."	"Void scaling for
Attribute Difference	Reference Attribute	Comparison Attrib
Access Attribute is different	rwrw	rwr-

Both workspaces have been changed. The comparison may not be up to date.

Status: Ready | Object: 12 (DD0) | User Mode | Access: rwrw | 0 Children

Use the **Refresh** button to update the comparison.

Method

To merge or replace a DD object using the DD two-way comparison

1. In the DD two-way comparison pane right-click the selected DD object to merge the **Reference: <root path>** into the **Comparison: <root path>** or vice versa.
2. Select the appropriate merge option:

Command
Merge Left/Right, and Overwrite Objects (refer to Merge <Left/Right/Middle>, and Overwrite Objects (TargetLink Data Dictionary Manager Reference))
Merge Left/Right, Without Overwrite (refer to Merge <Left/Right/Middle> Without Overwrite (TargetLink Data Dictionary Manager Reference))
Merge Left/Right, and Overwrite Properties (refer to Merge <Left/Right/Middle>, and Overwrite Properties (TargetLink Data Dictionary Manager Reference))
Merge Left/Right (refer to Merge <Left/Right/Middle> (TargetLink Data Dictionary Manager Reference))
Replace Left/Right (refer to Replace <Left/Right/Middle> (TargetLink Data Dictionary Manager Reference))

Note

You can also merge specific property values using the merge option in the context menus:

- If you select Merge Right in the Reference Value field, the Reference Value property is copied to the Comparison Value property.
- If you select Merge Left in the Comparison Value field, the Comparison Value property is copied to the Reference Value property.

Result

You merged or replaced a DD object using the DD two-way comparison.

Next Steps

To generate an HTML or XML comparison report, refer to How to Generate a Comparison Report Using the DD Two-Way Comparison on page 97.

Related topics

Basics

Merge <Left/Right/Middle> (📖 TargetLink Data Dictionary Manager Reference)

HowTos

References

Compare with <DD Object> (📖 TargetLink Data Dictionary Manager Reference)
dsddman('Compare', propertyName, propertyValue, ...) (📖 TargetLink API Reference)
Merge <Left/Right/Middle> Without Overwrite (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle>, and Overwrite Objects (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle>, and Overwrite Properties (📖 TargetLink Data Dictionary Manager Reference)
Replace <Left/Right/Middle> (📖 TargetLink Data Dictionary Manager Reference)

How to Generate a Comparison Report Using the DD Two-Way Comparison

Objective

To generate a comparison report in HTML or XML for the entire DD object tree using the DD two-way comparison.

Note

You can also use `DsddComp` in the Windows Command Prompt to compare two DD files and write the differences to a report. Change the directory to `<Installation Directory of TargetLink>\dsdd\bin` and use the following command:

```
DsddComp <reference file> <corresponding file>
-xmlreport <report file> [-settings <options file>]
```

Precondition

You compared DD objects using the DD two-way comparison. Refer to How to Compare DD Objects Using the DD Two-Way Comparison on page 93.

Restriction

It is not possible to generate an HTML report in the stand-alone mode of the DD Manager.

Method

To generate a comparison report using the DD two-way comparison

1. In the DD two-way comparison pane, right-click a DD object.
2. Select Generate report and choose either Xml (starting at Root) or Html (starting at Root).
3. Enter a name for the comparison report and click Save.
   - HTML report: The MATLAB® Web browser opens and shows the HTML comparison report.
   - XML report: An XML file is created and can be edited with any XML tool.

Result

You generated a comparison report in HTML or XML using the DD two-way comparison.

Related topics

HowTos

References

Compare with <DD Object> (📖 TargetLink Data Dictionary Manager Reference)
Data Dictionary Limitations (📖 TargetLink Limitation Reference)
dsddman('Compare', propertyName, propertyValue, ...) (📖 TargetLink API Reference)

Basics on Comparing and Merging DD Files via DD Three-Way Merge

Overview of the DD three-way merge

The DD three-way merge compares two DD files derived from a common ancestor DD file. For example, during a development project, a common ancestor DD project file can be used in two different development branches. The two derived DD project files, the reference and comparison DD files, are then compared with their common ancestor DD file via the DD three-way merge and merged into one DD workspace.

With the DD three-way merge, you can do the following:

- Identify changes in the reference and comparison DD file in comparison to the common ancestor.
- Identify conflicts between the reference and comparison DD file.
- Create a merged DD workspace that contains contents from the reference, comparison, and common ancestor DD file.

Note

If you want to compare two DD files regardless of their development branches, you can use the DD two-way comparison.
Refer to Basics on Comparing and Merging DD Objects and Workspaces Using the DD Two-Way Comparison on page 87.

Starting the DD three-way merge

You can start the DD three-way merge via the following methods:

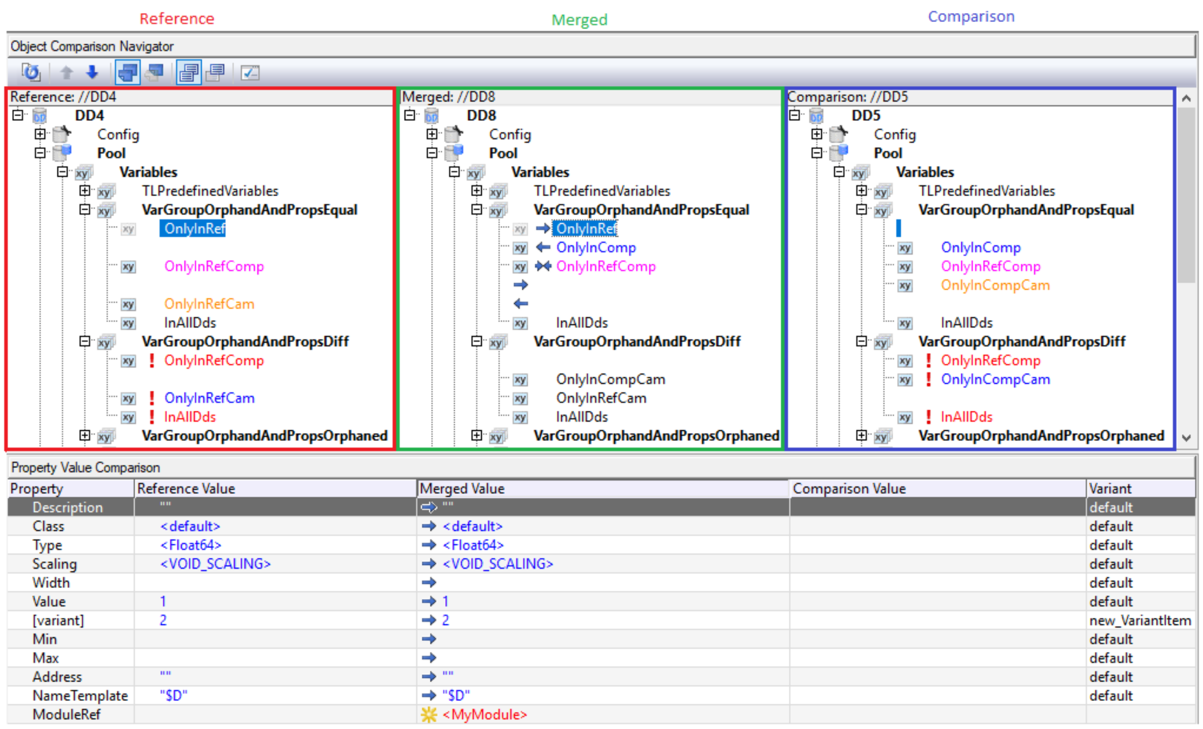
Method	Instructions
Windows Command Prompt	Change the directory to `<InstallationDir>\dsdd\bin` and use the `DsddMan.exe` command. To compare two DD project files with a common ancestor DD project file, use: [1)] `DsddMan -reffile <reffile> -compfile <compfile> -ancfile <ancfile> -mergedfile <mergedfile> [-reftitle <reftabtitle>] [-comptitle <comptabtitle>] [-mergedtitle <mergedtabtitle>] [-automerge on|off] [-settings <sfile>]`
Version control system	Refer to Basics on Configuring Version Control Systems to Compare and Merge DD Files on page 90.

1) Enter `DsddMan /?` to display a help window with a list of all possible command line arguments.

Note

The DD three-way merge initializes the merged DD workspace with the content of the common ancestor DD file. It automatically merges non-conflicting changes made in the reference or comparison DD file into the merged DD workspace (default). If you want to omit the automatic merging step, you have to set automerge to off via Windows Command Prompt or a version control system.

Working with the DD three-way merge



Property	Reference Value	Merged Value	Comparison Value	Variant
Description	""	⇨ ""		default
Class	<default>	→ <default>		default
Type	<Float64>	→ <Float64>		default
Scaling	<VOID_SCALING>	→ <VOID_SCALING>		default
Width		→		default
Value	1	→ 1		default
[variant]	2	→ 2		new_VariantItem
Min		→		default
Max		→		default
Address	""	→ ""		default
NameTemplate	"$D"	→ "$D"		default
ModuleRef		✱ <MyModule>		

The user interface and navigation in the DD three-way merge is similar to the DD two-way comparison. Refer to Basics on Comparing and Merging DD Objects and Workspaces Using the DD Two-Way Comparison on page 87 and Compare with <DD Object> (📖 TargetLink Data Dictionary Manager Reference).

Color coding and symbols of the DD three-way merge Colors indicate changes in the reference, comparison, or merged DD file:

Color	Description
Red	Properties of DD objects in the reference, comparison, or merged DD workspace were changed in comparison to the common ancestor.
Blue	The DD object was added to the reference, comparison, or merged DD workspace in comparison to the common ancestor.
Magenta	The DD object is identical in the reference, comparison, or merged DD workspace, but different from the common ancestor.
Orange	The DD object was deleted from the reference, comparison, or merged DD workspace in comparison to the common ancestor.

DD objects with differences in DD child objects are displayed in bold letters.

The following symbols can be displayed:

Symbol	Description
⬅	For the merged DD workspace, the values of the DD object or property of the reference DD workspace are taken into account.
➡	For the merged DD workspace, the values of the DD object or property of the comparison DD workspace are taken into account.
⇥⇤	For the merged DD workspace, values of the DD object or property of the comparison and reference DD workspace are taken into account.
✳	DD objects or property values that are manually added to the merged DD workspace and do not exist in either the comparison, reference, nor common ancestor DD workspace.
!	The red exclamation mark indicates DD objects or properties with conflicts that must be resolved manually.

Customizing the three-way analysis

Use the Data Dictionary Compare Tool Options dialog to customize your three-way analysis. Refer to Compare Tool Options (📖 TargetLink Data Dictionary Manager Reference).

Related topics

HowTos

How to Perform a DD Three-Way Merge

Objective

To perform a DD three-way merge of two DD files deriving from different development branches of a common ancestor DD file.

Method

To perform a DD three-way merge

1 Start the DD three-way merge via Windows Command Prompt or a version control system.

Refer to Basics on Comparing and Merging DD Files via DD Three-Way Merge on page 98.

2 Analyze the results of the DD three-way merge. The displayed colors and symbols help you identify conflicts.

Refer to Basics on Comparing and Merging DD Files via DD Three-Way Merge on page 98.

3 Right-click the conflicting DD object or property and select the appropriate merge option to solve the conflicts:

Command
Merge Middle, and Overwrite Objects (refer to Merge <Left/Right/Middle>, and Overwrite Objects (📖 TargetLink Data Dictionary Manager Reference))
Merge Middle, Without Overwrite (refer to Merge <Left/Right/Middle> Without Overwrite (📖 TargetLink Data Dictionary Manager Reference))
Merge Middle, and Overwrite Properties (refer to Merge <Left/Right/Middle>, and Overwrite Properties (📖 TargetLink Data Dictionary Manager Reference))
Merge Middle (refer to Merge <Left/Right/Middle> (📖 TargetLink Data Dictionary Manager Reference))
Replace Middle (refer to Replace <Left/Right/Middle> (📖 TargetLink Data Dictionary Manager Reference))
Comfort Copy Middle (refer to Comfort Copy Middle (📖 TargetLink Data Dictionary Manager Reference))

4 Save the merged DD workspace to a DD file.

Result

You performed a DD three-way merge.

Related topics

Basics

Basics on Comparing and Merging DD Files via DD Three-Way Merge.................................... 98
Merge <Left/Right/Middle> (📖 TargetLink Data Dictionary Manager Reference)

References

Comfort Copy Middle (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle> Without Overwrite (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle>, and Overwrite Objects (📖 TargetLink Data Dictionary Manager Reference)
Merge <Left/Right/Middle>, and Overwrite Properties (📖 TargetLink Data Dictionary Manager Reference)
Replace <Left/Right/Middle> (📖 TargetLink Data Dictionary Manager Reference)

How to Generate a Comparison Report Using the DD Three-Way Merge

Objective

To generate a comparison report in XML using the DD three-way merge.

Precondition

You merged DD objects using the DD three-way merge. Refer to How to Perform a DD Three-Way Merge on page 101.

Method

To generate a comparison report using the DD three-way merge

1. In the DD three-way merge pane, right-click a DD object.
2. Select Generate Xml report (Starting at Root).
3. Enter a name for the comparison report and click Save.
   An XML file of the complete comparison is created and can be edited with any XML tool.

Result

You generated a comparison report in XML using the DD three-way merge.

Related topics

Basics

HowTos

Inspecting and Handling Multiple Data Dictionary Objects

Where to go from here

Information in this section

Basics on Handling Multiple DD Objects with the Object Explorer

Introduction

You sometimes need to inspect and set the property values of various DD objects, for example, to inspect identical variable classes at several Variable objects. You can do this in the Object Explorer.

Object Explorer

In the Data Dictionary Manager, the Object Explorer displays DD objects and their properties in a table with the objects in rows and the properties in columns.

You can show or hide the Object Explorer if necessary. When you close the Data Dictionary Manager, the state of the Object Explorer is automatically saved. It is restored the next time the DD Manager is opened.

The Object Explorer lets you select several DD objects, to assign property values to them simultaneously.

Specifying objects and properties

You can specify the DD objects and the properties to be displayed in the Object Explorer via the toolbar, the Filter and Depth lists, and the Property Selector.

Toolbar The toolbar offers the option to show and hide additional columns in the Object Explorer.

Filter The settings in the Filter list specify which object types are listed in the Object Explorer.

Depth The settings in the Depth list specify which level of child objects is displayed in the Object Explorer.

Property Selector The settings in the Property Selector determine which properties are shown. Properties which do not exist at a DD object are disabled (grayed out) in the respective row. The Property Selector can be displayed only if the Object Explorer is displayed.

Tip

Additionally, you can add properties to the Object Explorer by dragging them from the Property Value List, or via context menu.

Editing properties of multiple DD objects

For a better overview, you can sort the objects in the Object Explorer by clicking the column header of any of the displayed properties.

The Object Explorer provides several ways to edit the properties of the selected DD objects. You can:

- Enter a value directly in the respective field.
- Select predefined values from a list.
- Open a appropriate dialog.

How to Specify Objects Shown in the Object Explorer

Objective

To inspect and edit the properties of multiple Data Dictionary objects, you have to specify the objects to be displayed in the Object Explorer.

Preconditions

The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.

Method

To specify objects shown in the Object Explorer

1. If the Object Explorer is not visible, you must open it via the View - Show Object Explorer menu command or Toolbar button.
2. In the Data Dictionary Navigator, expand the DD object tree and select the DD object you want to display in the Object Explorer.
3. From the Filter list, select the object kind to be displayed.
4. From the Depth list, select the level of child objects to be displayed.

Result

The specified DD objects are displayed in the Object Explorer.

Next steps

After you have specified DD objects to be shown in the **Object Explorer**, you can specify the properties (columns) to be displayed. Refer to How to Specify Properties (Columns) Shown in the Object Explorer on page 106.

Related topics

References

Data Dictionary Navigator (📖 TargetLink Data Dictionary Manager Reference)
Make Primary DD Workspace (📖 TargetLink Data Dictionary Manager Reference)
Object Explorer (📖 TargetLink Data Dictionary Manager Reference)
Object Explorer - Depth (📖 TargetLink Data Dictionary Manager Reference)
Object Explorer - Filter (📖 TargetLink Data Dictionary Manager Reference)
Object Explorer - Show Add/Remove Column (📖 TargetLink Data Dictionary Manager Reference)
Object Explorer - Show All Selected Properties (📖 TargetLink Data Dictionary Manager Reference)
Show DD Workspace Overview (📖 TargetLink Data Dictionary Manager Reference)
Show Object Explorer (📖 TargetLink Data Dictionary Manager Reference)

How to Specify Properties (Columns) Shown in the Object Explorer

Objective

To inspect and edit the properties of multiple Data Dictionary objects, you have to specify the properties (columns) to be displayed in the **Object Explorer**.

Preconditions

The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.

Possible methods

You can specify the properties (columns) shown in the **Object Explorer** by using one or a combination of the following methods:

- You can select properties via the **Property Selector**. Refer to Method 1.
- You can add properties by selecting one or multiple DD objects in the **Object Explorer** and then selecting the checkboxes in the **Property Value List**. Refer to Method 2.
- You can add properties to the **Object Explorer** by dragging them from the **Property Value List**. Refer to Method 3.
- You can add properties to the **Object Explorer** via the context menu of the **Property Value List**. Refer to Method 4.
- You can show properties which are selected in the **Property Selector** but do not exist at the displayed DD objects. Refer to Method 5.

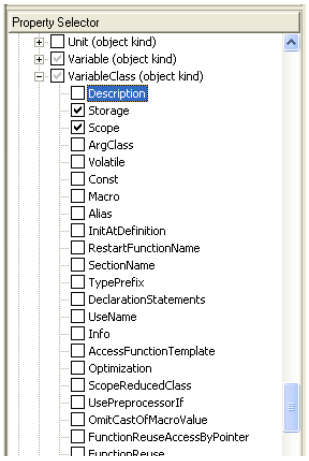
Method 1

To specify properties to be shown via the Property Selector

1 If the Object Explorer is not visible, you must open it via the View - Show Object Explorer menu command or Toolbar button.

2 If the Property Selector is not visible, you must open it via the View - Show Property Selector menu command or Toolbar button.

3 In the Property Selector, expand the tree and select the properties.


Property Selector
- ☐ Unit (object kind)
- ☐ Variable (object kind)
- ☐ VariableClass (object kind)
  - ☐ Description
  - ☑ Storage
  - ☑ Scope
  - ☐ ArgClass
  - ☐ Volatile
  - ☐ Const
  - ☐ Macro
  - ☐ Alias
  - ☐ InitAtDefinition
  - ☐ RestartFunctionName
  - ☐ SectionName
  - ☐ TypePrefix
  - ☐ DeclarationStatements
  - ☐ UseName
  - ☐ Info
  - ☐ AccessFunctionTemplate
  - ☐ Optimization
  - ☐ ScopeReducedClass
  - ☐ UsePreprocessorIf
  - ☐ OmitCastOfMacroValue
  - ☐ FunctionReuseAccessByPointer
  - ☐ FunctionReuse

Method 2

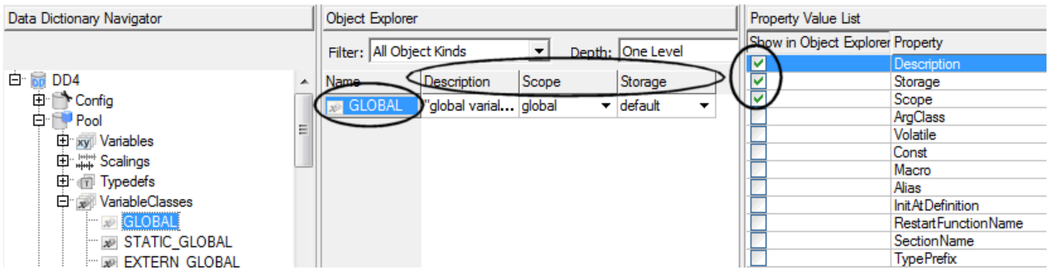
To add properties (columns) from the Property Value List to the Object Explorer by selecting checkboxes

1 If the Object Explorer is not visible, you must open it via the View - Show Object Explorer menu command or Toolbar button.

2 Select one or more objects in the Object Explorer.

In the Property Value List, a checkbox for each property appears.

3 You can add or remove properties (columns) by selecting or clearing the checkboxes in Show in Object Explorer.



Name	Description	Scope	Storage
GLOBAL	"global varial...	global	default

Property Value List

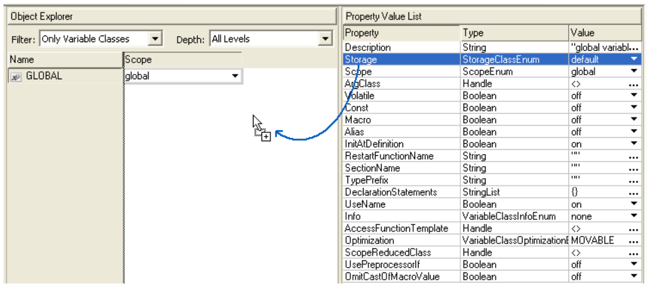
Show in Object Explorer	Property
☑	Description
☑	Storage
☑	Scope
☐	ArgClass
☐	Volatile
☐	Const
☐	Macro
☐	Alias
☐	InitAtDefinition
☐	RestartFunctionName
☐	SectionName
☐	TypePrefix

Method 3

To add properties (columns) from the Property Value list to the Object Explorer by drag & drop

1 If the Object Explorer is not visible, you must open it via the View - Show Object Explorer menu command or Toolbar button.

2 Drag the property from the Property Value List.
A vertical line indicates the position where the new column is inserted.



Name	Scope
GLOBAL	global

Property	Type	Value
Description	String	"global variabl...
Storage	StorageClassEnum	default
Scope	ScopeEnum	global
ArgClass	Handle	<>
Volatile	Boolean	off
Const	Boolean	off
Macro	Boolean	off
Alias	Boolean	off
InitAtDefinition	Boolean	on
RestartFunctionName	String	""
SectionName	String	""
TypePrefix	String	""
DeclarationStatements	StringList	{}
UseName	Boolean	on
Info	VariableClassInfoEnum	none
AccessFunctionTemplate	Handle	<>
Optimization	VariableClassOptimizationE	MOVABLE
ScopeReducedClass	Handle	<>
UsePreprocessorIf	Boolean	off
OmitCastOfMacroValue	Boolean	off

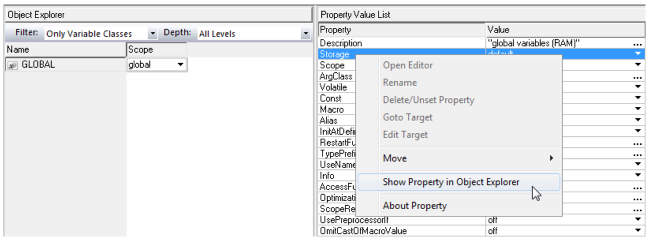
3 Drop the property in the Object Explorer.

Method 4

To add properties (columns) via the context menu of the Property Value List

1 If the Object Explorer is not visible, you must open it via the View - Show Object Explorer menu command or Toolbar button.

2 In the Property Value List, right-click the property.



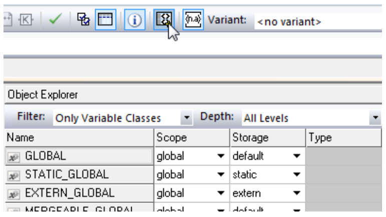
3 From the context menu, select Show Property in Object Explorer.

Method 5

To show properties which do not exist at the displayed DD objects

1 If the Object Explorer is not visible, you must open it via the View - Show Object Explorer menu command or Toolbar button.

2 Click the Show all Selected Properties button.



Name	Scope	Storage	Type
GLOBAL	global	default	
STATIC_GLOBAL	global	static	
EXTERN_GLOBAL	global	extern	
MERGEABLE_GLOBAL	global	default	

Result

The specified properties (columns) are shown in the Object Explorer.

Next steps

After you have specified properties (columns) to be shown in the **Object Explorer**, you can set the properties of multiple Data Dictionary objects. Refer to How to Set the Properties of Multiple Data Dictionary Objects on page 111.

Related topics

References

Object Explorer (📖 TargetLink Data Dictionary Manager Reference)
Object Explorer - Show All Selected Properties (📖 TargetLink Data Dictionary Manager Reference)
Property Selector (📖 TargetLink Data Dictionary Manager Reference)
Property Value List (📖 TargetLink Data Dictionary Manager Reference)
Show Object Explorer (📖 TargetLink Data Dictionary Manager Reference)
Show Property in Object Explorer (📖 TargetLink Data Dictionary Manager Reference)
Show Property Selector (📖 TargetLink Data Dictionary Manager Reference)

How to Sort DD Objects by Property

Objective

To inspect and edit the properties of multiple Data Dictionary objects, it is advisable to sort the objects by a specific property.

Preconditions

To perform the steps described below, the following preconditions must be fulfilled:

- The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.
- The **Object Explorer** is displayed.

Possible methods

You can sort the objects by property by using one of the following methods:

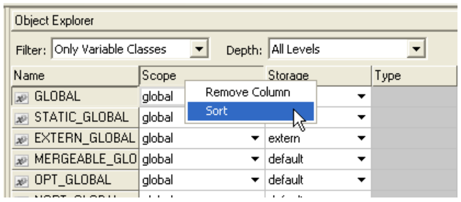
- You can select a command from a context menu. Refer to Method 1.
- You can click a column header. Refer to Method 2.

Method 1

To sort the DD objects in the Object Explorer by a property

1 Right-click the header of the column you want to sort the objects by.

A context menu opens.



Object Explorer

Filter: Only Variable Classes Depth: All Levels

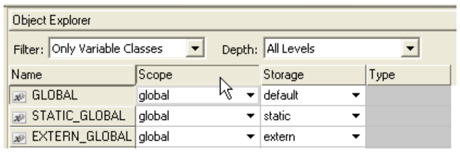
Name	Scope	Storage	Type
GLOBAL	global		
STATIC_GLOBAL	global		
EXTERN_GLOBAL	global	extern	
MERGEABLE_GLO	global	default	
OPT_GLOBAL	global	default	

2 From the context menu, select Sort.

Method 2

To sort the DD objects in the Object Explorer by a property

1 Click the header of the column you want to sort the objects by.



Object Explorer

Filter: Only Variable Classes Depth: All Levels

Name	Scope	Storage	Type
GLOBAL	global	default	
STATIC_GLOBAL	global	static	
EXTERN_GLOBAL	global	extern	

Result

The DD objects in the Object Explorer are sorted by the selected property.

Tip

To undo the sorting, you can click the DD object tree in the Data Dictionary Navigator.

Next steps

You can set the properties of multiple Data Dictionary objects. Refer to How to Set the Properties of Multiple Data Dictionary Objects on page 111.

Related topics

References

Object Explorer (📖 TargetLink Data Dictionary Manager Reference)

Example of Inspecting the Properties of Multiple Data Dictionary Objects

Introduction

The Data Dictionary Manager's Object Explorer allows you to inspect the properties of multiple DD objects simultaneously. This gives you an overview of the DD objects contained in a DD project files and their properties.

Preconditions

To perform the steps described below, the following preconditions must be fulfilled:

- The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.

Method

To inspect the properties of multiple DD objects

1. In the toolbar, click the Show all Selected Properties button.
   All columns displayed in the Object Explorer are hidden.
2. In the Property Selector, select All Properties.
3. In the Data Dictionary Navigator, select the `/Config` object.
4. From the Filter list, select All Object Kinds.
5. From the Depth list, select All Levels.

Result

In the Object Explorer, all objects under the `/Config` object are displayed with their properties.

Tip

Select the `/Config/Units` object in the Data Dictionary Navigator to get an overview of the physical units.

Related topics

References

Object Explorer (📖 TargetLink Data Dictionary Manager Reference)
Object Explorer - Show All Selected Properties (📖 TargetLink Data Dictionary Manager Reference)

How to Set the Properties of Multiple Data Dictionary Objects

Objective

The Data Dictionary Manager's Object Explorer allows you to set identical properties for multiple DD objects simultaneously.

Preconditions

To perform the steps described below, the following preconditions must be fulfilled:

- The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.

- You have specified the objects and properties to be shown in the Object Explorer. Refer to How to Specify Objects Shown in the Object Explorer on page 105 and How to Specify Properties (Columns) Shown in the Object Explorer on page 106

Possible methods

You can set the properties of multiple DD objects by using one of the following methods:

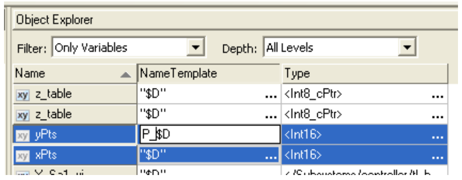
- You can enter values in edit fields. Refer to Method 1.
- You can select values from lists. Refer to Method 2.
- You can set properties via dialogs. Refer to Method 3.

Method 1

To set the properties of multiple DD objects by entering a value in a property field

1 In the Object Explorer, select the DD objects you want to edit.

2 In an edit field of the property (column), enter the new value.


Object Explorer

Filter: Only Variables　Depth: All Levels

Name	NameTemplate	Type
z_table	"$D" ...	<Int8_cPtr> ...
z_table	"$D" ...	<Int8_cPtr> ...
yPts	P_$D	<Int16> ...
xPts	"$D" ...	<Int16> ...

3 Press Enter.

Method 2

To set the properties of multiple DD objects by selecting a value from a list

1 In the Object Explorer, select the DD objects you want to edit.

2 From a drop-down list for the property, select the new value.

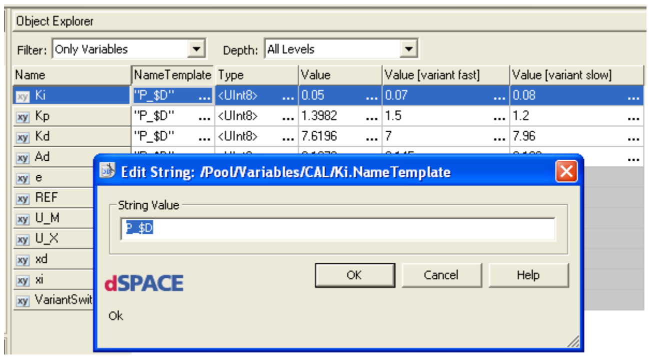
3 Press Enter.

Method 3

To set the properties of multiple DD objects via a dialog

1 In the Object Explorer, select the DD objects you want to edit.

2 In an edit field of the property, click the Edit button.
  The appropriate dialog opens.



Object Explorer

Filter: Only Variables Depth: All Levels

Name	NameTemplate	Type	Value	Value [variant fast]	Value [variant slow]
Ki	"P_$D" ...	<UInt8> ...	0.05 ...	0.07 ...	0.08 ...
Kp	"P_$D" ...	<UInt8> ...	1.3982 ...	1.5 ...	1.2 ...
Kd	"P_$D" ...	<UInt8> ...	7.6196 ...	7 ...	7.96 ...
Ad	[illegible]	[illegible]	[illegible]	[illegible]	[illegible] ...
e					
REF					
U_M					
U_X					
xd					
xi					
VariantSwit					

Edit String: /Pool/Variables/CAL/Ki.NameTemplate

String Value: P_$D

Ok

3 Enter the new value.

4 Click OK.

Result

You have set the properties of the selected DD objects.

Related topics

HowTos

How to Multi-Edit Property Values (📖 TargetLink Preparation and Simulation Guide)

References

Object Explorer (📖 TargetLink Data Dictionary Manager Reference)

Managing Data with the DD MATLAB API

Introduction

The TargetLink Data Dictionary MATLAB API is an application programming interface that lets you access the TargetLink Data Dictionary via MATLAB. You can then handle the Data Dictionary via MATLAB API functions.

Where to go from here

Information in this section

Getting Started with the DD MATLAB® API

Basics on the DD MATLAB® API

Accessing the TargetLink Data Dictionary via API

You can access the TargetLink Data Dictionary (DD) via the MATLAB application programming interface (API) to automate the following tasks, for example:

- Creating and specifying DD objects and properties.
- Modifying and deleting DD objects and properties.
- Searching and validating DD objects and properties.
- Importing or exporting data to or from the DD.
- Managing DD workspaces.

Using the DD MATLAB API

You can use the DD MATLAB API directly from the MATLAB Command Window. You can also create M files to run a sequence of commands.

Example

```
% Add a new DD Variable object
dsdd('AddVariableObject','/Pool/Variables','Kp');
% Set the Class property to CAL
dsdd('SetClass','/Pool/Variables/Kp','CAL');
% Set the Type property to UInt16
dsdd('SetType','/Pool/Variables/Kp','UInt16');
```

Demo models

The following demo models introduce you the DD MATLAB API:

- DD_ML_API (TargetLink Demo Models)
- DD_ML_IMPORTEXPORT (TargetLink Demo Models)

Related topics

Basics

Basics on Handling Data Dictionary Objects via API.. 117
DD_ML_API (TargetLink Demo Models)
DD_ML_IMPORTEXPORT (TargetLink Demo Models)

References

Generic Commands (TargetLink Data Dictionary Reference)
Object-Specific Commands (TargetLink Data Dictionary Reference)

Managing Data Dictionary Content via API

Where to go from here

Information in this section

Basics on Handling Data Dictionary Objects via API

Handling DD objects via API

You can use two different command sets to handle DD objects and properties:

- Object-specific commands
- Generic commands

Object-specific commands Object-specific commands handle DD objects and properties which are defined in the data model. You can use object-specific commands for modeling purposes, for example, to add new DD objects and set properties.

Object-specific commands help you reduce specification errors because they return error messages if you access properties that are not defined in the data model.

```
% Add a new DD Variable object
dsdd('AddVariableObject','/Pool/Variables','Kp');
% Set the Class property to CAL
dsdd('SetClass','/Pool/Variables/Kp','CAL');
% Set the Type property to UInt16
dsdd('SetType','/Pool/Variables/Kp','UInt16');
```

The object-specific commands usually have the following naming conventions:

```
dsdd('Set<PropertyName>', ...)
dsdd('Get<PropertyName>', ...)
dsdd('Add<ObjectOrPropertyName>', ...)
dsdd('Create<ObjectOrPropertyName>', ...)
```

In addition, you can list for all object-specific commands for a DD object as follows:

```
% List all object-specific commands for the DD Variable object in
%   in /Pool/Variables/new_Variable
dsdd('','/Pool/Variables/new_Variable');
```

Refer to Object-Specific Commands.

Generic commands Generic commands handle user-defined objects, custom properties, and metadata, for example, object paths and names for which no object-specific commands exist. You can use generic commands to copy data from a database or the MATLAB workspace to the DD workspace.

```
% Create a user-defined DD object
dsdd('Create', '/MyUserData');
% Set custom properties
dsdd('Set', '/MyUserData', ...
    'Owner', 'EGR', ...
    'Data', [1 2 3 4 5], ...
    'CompanyProp', pi);
```

Refer to Generic Commands.

Checking DD objects and properties via API

You can use generic commands to check DD objects and properties as follows:

```
% Find all DD Variable objects in /Pool/Variables whose
%   Class property is STATIC_CAL
dsdd('Find', '/Pool/Variables', ...
    'ObjectKind', 'Variable', ...
    'Property', {'name' 'Class' 'value' 'STATIC_CAL'});
% Get all DD properties for the DD object in /Pool/Variables/Position
dsdd('GetAll','/Pool/Variables/Position');
```

Validating DD objects via API

You can use a generic command to validate DD objects in a specified tree.

```
% Validate all DD objects in the /Config and /Pool area
dsdd_validate({'/Config','/Pool'},'Level',4);
```

Related topics

References

dsdd_validate (TargetLink API Reference)
Generic Commands (TargetLink Data Dictionary Reference)
Object-Specific Commands (TargetLink Data Dictionary Reference)

Basics on Managing Data Dictionaries with the DD MATLAB API

Introduction

The TargetLink Data Dictionary works with project files which contain project-related data. When a project is opened, the respective project file is loaded into the data dictionary.

Until the DD project file is saved, any changes to the project are present only in the Data Dictionary memory. Saving the project writes the changes to the DD project file. The MATLAB API provides you with a set of commands for managing the data dictionary. For detailed information on these commands, refer to the following commands in the TargetLink Data Dictionary Reference.

Retrieving DD workspace information

- To retrieve information on a specific data dictionary workspace, use GetDDAttribute or GetDDAttributes.
- To check if the DD workspace has been modified since it was last opened or saved, use IsModified.
- To check if the raw mode is currently switched on or off, use GetRawMode.

Setting DD workspace information

- To delete all the objects in the specified DD workspace, use Clear.
- To switch to the raw mode, use SetRawMode.

Note

In raw mode, all checks for Data Model compatibility are disabled. The raw mode is mainly intended for DD file upgrade or downgrade purposes.

Example

```
% get info about workspace DD0
ddAttributes = dsdd('GetDDAttributes','DD0');
% check if the raw mode is enabled
rm = dsdd('GetRawMode')
% clear all objects in DD0
dsdd('Clear')
```

Basics on Handling Data Dictionary Files in the DD MATLAB API

Introduction

The TargetLink Data Dictionary saves the DD workspace to a DD project file. The MATLAB API provides you with a set of commands for handling partial DD files or whole DD project files. You can open and close projects files or save and load parts of the DD workspace. Additionally, you can export files with different data formats from the data dictionary or import them to the data dictionary.

For detailed information on the commands for working with data files, refer to the following commands in the TargetLink Data Dictionary Reference:

Working with the whole project file

- To open a project file, use Open.
- To close a project file, use Close.

Working with parts of a project file

- To open a partial DD file or a full DD project file, use AutoLoad.
- To save a part of the DD object tree as a DD file, use Save.
- To load a DD file into an object in the DD object tree, use Load.
- To export a part of the DD object tree to file in a different data format, use Export.
- To retrieve attributes of a (partial or project) DD file, use GetFileAttributes.
- To import data in a different data format into the data dictionary, use Import.

Example

```
% open DD project file
dsdd('Open','default.dd')
% create a new Variable object
dsdd('AddVariableObject','/Pool/Variables','MyVar')
% save to file
dsdd('Save','//DD0','File','default_new.dd')
% save all variables
dsdd('Save','/Pool/Variables','File','my_variables.dd')
% export /Pool/Variables as an XML file
dsdd('Export','Format','xml','File','myproject.xml','mode','extended', ...
    'Root','/Pool/Variables')
% open a DD file into workspace DD8 for inspection
dsdd('AutoLoad', 'file', 'someFile.dd', 'DDIdx', 8)
```

How to Associate a DD Project File with a TargetLink Model via API

Precondition

The DD project file you want to associate is on the current MATLAB® search path.

Method

To associate a DD project file with a TargetLink model via API

1 Associate a DD project file with a TargetLink model.

```
dsdd_manage_project('SetProjectFile', <ProjectFile>, <ModelName>);
```

Tip

Activate batch mode for batch processing.

Result

You associated a DD project file with a TargetLink model via API.

Related topics

Basics

Basics on the Data Dictionary Manager.. 31

References

dsdd_manage_project('SetProjectFile', projectFile, simulinkSystem) (TargetLink API Reference)

Basics on Filter Rule Sets for the Data Model

Introduction

To customize views for different team members, the Data Dictionary enables to filter objects and properties displayed in the DD Manager by xml-based filter rule sets. Filter rules allow you to have specified objects and properties *not displayed* in the Data Dictionary Manager, the Property Manager, and the DD Reference Selection dialog. In addition, you can even use filter rules in the Object Comparison Navigator if you want to filter compared DD objects and properties. This helps you to focus on objects that are of interest for the current use case, for example, Modeling.

Filter rules and filter rule sets

A filter rule specifies a path in the Data Model were the contained objects and properties are hidden. A filter rule set is an aggregate of filter rules, identified by a keyword. A filter rule set can be written to one or several XML files.

Default and predefined filter rule sets

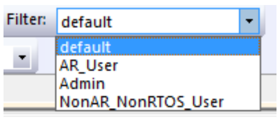
There is always a default filter rule set. This is the set which applies when the DD starts before any other set is selected. The default filter shows all relevant objects and properties. In order to hide unnecessary objects/properties for your specific use case, you can create your own filter rules. Refer to How to Create Filter Rule Sets via DD Files on page 123 and How to Create Filter Rule Sets via API Functions on page 126.

In addition, the following predefined filter rule sets are available:

- AR_User - A filter rule set for AUTOSAR users.
- Admin - a filter rule set adapted for DD administrators.
- NonAR_NonRTOS_User - A filter rule set for both non-AUTOSAR and non-RTOS users. In addition, some internal objects (of no interest for normal users) are hidden, too.

Selecting a filter rule set

You can select a filter rule set in the Filter list of the Data Dictionary Manager's toolbar.



As an alternative, use the API function SetCurrentFilterRuleSet.

Note

With the DD MATLAB API functions, all objects can be accessed independently of the currently selected user-defined view.

Creating filter rule sets

You can create your own filter rule sets in different ways:

- An easy way is to use the `tl_example_CreateDDFilterBasedOnDDFile` script to create an XML filter rule set based on a DD file containing the filter definition (for example, visible and invisible data model paths). Refer to How to Create Filter Rule Sets via DD Files on page 123.
- You can create filter rule sets using API functions. Refer to How to Create Filter Rule Sets via API Functions on page 126.

Activating new filter rule sets

To activate new filter rule sets so that they become visible in the Filter list, refer to How to Activate New Filter Rule Sets on page 125.

Additional examples

For additional information on creating filter rule sets, refer to Examples of Filter Rule Sets for Different Use Cases on page 127

Related topics

HowTos

Examples

How to Create Filter Rule Sets via DD Files

Introduction

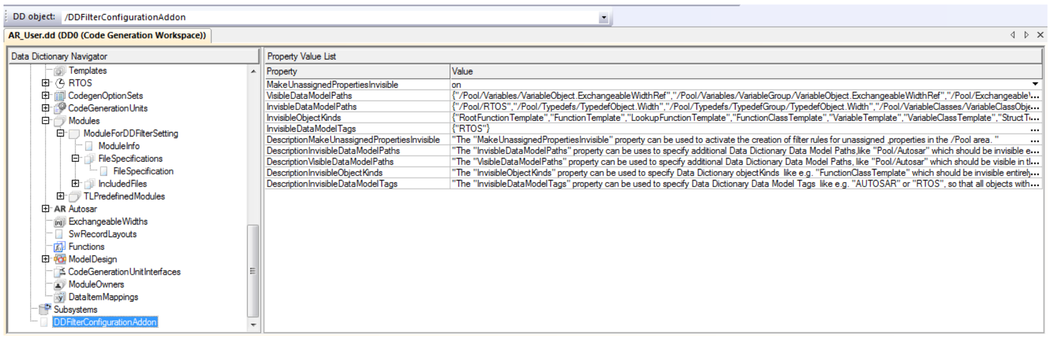
You can create filter rule sets for the Data Dictionary Manager to focus on relevant objects of a specific use case. The DD_Filter TargetLink demo comes with an M script that opens a DD file containing the filter specifications and generates an XML file containing the filter rule set. Then you add the filter to the DD Manager by activation (see below).

Precondition

You must have specified filter data (for example, visible and invisible data model paths) in a DD file. As an alternative, you can use the predefined DD files that already contain filter data. You can change the specifications of the DD files according to your needs. In `<User>\<DocumentsFolder>\dSPACE\TargetLink\<Version>\Demos\DD_Filter`, refer to

- `Admin.dd`
- `AR_User.dd`
- `NonAR_NonRTOS_User.dd`

The following illustration shows an extract of the filter data from `AR_User.dd:`



Property	Value
MakeUnassignedPropertiesInvisible	on
VisibleDataModelPaths	{"/Pool/Variables/VariableObject.ExchangeableWidthRef","/Pool/Variables/VariableGroup/VariableObject.ExchangeableWidthRef","/Pool/Exchangeable\...
InvisbleDataModelPaths	{"/Pool/RTOS","/Pool/Typedefs/TypedefObject.Width","/Pool/Typedefs/TypedefGroup/TypedefObject.Width","/Pool/VariableClasses/VariableClassObje...
InvisibleObjectKinds	{"RootFunctionTemplate","FunctionTemplate","LookupFunctionTemplate","FunctionClassTemplate","VariableTemplate","VariableClassTemplate","StructT...
InvisibleDataModelTags	{"RTOS"} ...
DescriptionMakeUnassignedPropertiesInvisible	"The "MakeUnassignedPropertiesInvisible" property can be used to activate the creation of filter rules for unassigned ,properties in the /Pool area. " ...
DescriptionInvisibleDataModelPaths	"The "InvisbleDataModelPaths" property can be uses to specify additional Data Dictionary Data Model Paths,like "Pool/Autosar" which should be invisible e...
DescriptionVisibleDataModelPaths	"The "VisibleDataModelPaths" property can be used to specify additional Data Dictionary Data Model Paths, like "Pool/Autosar" which should be visible in tl...
DescriptionInvisibleObjectKinds	"The "InvisibleObjectKinds" property can be used to specify Data Dictionary objectKinds like e.g. "FunctionClassTemplate" which should be invisible entirely...
DescriptionInvisibleDataModelTags	"The "InvisibleDataModelTags" property can be used to specify Data Dictionary Data Model Tags like e.g. "AUTOSAR" or "RTOS", so that all objects with...

Method

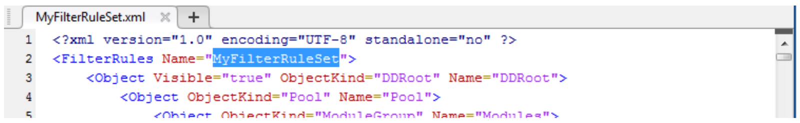
To create a filter rule set with a DD file

1 In the MATLAB Command Window, type `tl_demos dd_filter` to run the `tl_example_CreateDDFilterBasedOnDDFile` M script.

2 You are prompted to open a DD file containing the filter data. You can also use one of the predefined ones.

3 After the script has generated the filter rule set, you are prompted to save the XML file.

4 The filter rule set shown in the Filter list of the Data Dictionary Manager has the name of the source DD file. If you want to rename the filter rule set, open the generated XML file and change the name as shown in the following screenshot:



```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FilterRules Name="MyFilterRuleSet">
    <Object Visible="true" ObjectKind="DDRoot" Name="DDRoot">
        <Object ObjectKind="Pool" Name="Pool">
            <Object ObjectKind="ModuleGroup" Name="Modules">
```

5 Ensure that the filter rule set is saved in the right folder. Use the TargetLink Preferences dialog under Data Dictionary - Filter Rules or the `tl_pref('set', 'ddfilterruledirectory', <Directory>)` API function to set the preferences accordingly.

6 Type `dsdd('ReloadFilterRuleSets')` to ensure that the filters become visible in the Data Dictionary Manager.

Result

You have created a filter rule set with a DD file.

Related topics

Basics

HowTos

Examples

How to Activate New Filter Rule Sets

Introduction

You must activate new filter rule sets so that they become visible in the Filter list of the Data Dictionary Manager.

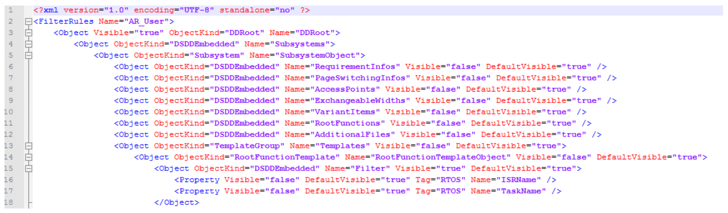
Precondition

You must have specified filter data (for example, visible and invisible data model paths) in an XML filter rule set file. Ensure that either the Use application data directory checkbox in the TargetLink Preferences Editor's Data Dictionary / Filter Rules pane is selected or another directory for the filter rule files is specified.

As an alternative, you can use the predefined XML filter rule files. You can change the visible and invisible data model paths according to your needs. In `<User>\<DocumentsFolder>\dSPACE\TargetLink\<Version>\Demos\DD_Filter\DD_Filter_XML_Files` (default setting after first start of `tl_demos`), refer to

- `Admin.xml`
- `AR_User.xml`
- `NonAR_NonRTOS_User.xml`

The following illustration shows an extract of the filter data from the `AR_User.xml`



```xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FilterRules Name="AR_User">
    <Object Visible="true" ObjectKind="DDRoot" Name="DDRoot">
        <Object ObjectKind="DSDDEmbedded" Name="Subsystems">
            <Object ObjectKind="Subsystem" Name="SubsystemObject">
                <Object ObjectKind="DSDDEmbedded" Name="RequirementInfos" Visible="false" DefaultVisible="true" />
                <Object ObjectKind="DSDDEmbedded" Name="PageSwitchingInfos" Visible="false" DefaultVisible="true" />
                <Object ObjectKind="DSDDEmbedded" Name="AccessPoints" Visible="false" DefaultVisible="true" />
                <Object ObjectKind="DSDDEmbedded" Name="ExchangeableWidths" Visible="false" DefaultVisible="true" />
                <Object ObjectKind="DSDDEmbedded" Name="VariantItems" Visible="false" DefaultVisible="true" />
                <Object ObjectKind="DSDDEmbedded" Name="RootFunctions" Visible="false" DefaultVisible="true" />
                <Object ObjectKind="DSDDEmbedded" Name="AdditionalFiles" Visible="false" DefaultVisible="true" />
                <Object ObjectKind="TemplateGroup" Name="Templates" Visible="false" DefaultVisible="true">
                    <Object ObjectKind="RootFunctionTemplate" Name="RootFunctionTemplateObject" Visible="false" DefaultVisible="true">
                        <Object ObjectKind="DSDDEmbedded" Name="Filter" Visible="true" DefaultVisible="true">
                            <Property Visible="false" DefaultVisible="true" Tag="RTOS" Name="ISRName" />
                            <Property Visible="false" DefaultVisible="true" Tag="RTOS" Name="TaskName" />
                        </Object>
```

Method

To activate a new filter rule set

1. Ensure that the XML filter rule set file is saved in the right folder. Use the TargetLink Preferences dialog under Data Dictionary - Filter Rules or the `tl_pref('set', 'ddfilterruledirectory', <Directory>)` API function to set the preferences accordingly.
2. Type `dsdd('ReloadFilterRuleSets')` to ensure that the filters become visible in the Data Dictionary Manager.

Result

You have activated a new filter rule set.

Related topics

Basics

HowTos

Examples

How to Create Filter Rule Sets via API Functions

Introduction

You can create filter rule sets for the Data Dictionary Manager to focus on relevant objects. You can create a filter rule set manually by using API functions.

Method

To create a filter rule set via API functions

1 In the Data Dictionary MATLAB API , create a filter rule set with the CreateFilterRuleSet command. The set should have a reasonable name which identifies it.

2 Set the current filter rule set with the SetCurrentFilterRuleSet command.

3 Specify filter rules with the SetFilterRule or SetFilterRuleByDataModelTag commands.

Note

The GetDataModelPath, GetDataModelPaths and DumpDataModelPaths commands are helpful to find the associated Data Model paths.

4 Save the filter rule set to an XML file with the WriteFilterRuleSet command.

5 Copy the filter rule file to the directory specified in TargetLink's preferences.

```
copyfile('MyFilterRuleSet.xml,
tl_pref('get', 'ddfilterruledirectory'));
```

Note

Ensure that either the **Use application data directory** checkbox in the TargetLink Preferences Editor's **Data Dictionary / Filter Rules** pane is selected or another directory for the filter rule files is specified.

Additional commands for filter rule sets

The following commands help you to manage filter rule sets:

- You can reset all filter rule sets with the **ResetFilterRuleSet** command
- Delete filter rule sets with the **DeleteFilterRuleSet** command
- Check filter rule sets with the **IsVisible** command in order to check if hidden objects and properties are not visible in the Data Dictionary Manager.

Result

You have created a new filter rule set via API functions.

Related topics

Examples of Filter Rule Sets for Different Use Cases

Introduction

In the following, some examples of filter rule sets for different use cases are provided.

Example 1: You want to hide specific objects and properties

You want to have a filter rule set where objects and properties related to Variable Vector Width (VVW) code generation are invisible.

```
% create filter rule set "NoVVW",
% and make it the current filter rule set
dsdd('CreateFilterRuleSet','NoVVW');
dsdd('SetCurrentFilterRuleSet','NoVVW');
```

```
% make all VVW objects and property invisible
dsdd('SetFilterRuleByDataModelTag','VVW','off');
% save the set to file - the filename is arbitrary
dsdd('WriteFilterRuleSet','file','NoVVW.xml');
```

The `NoVVW.xml` file now specifies the Variable Vector Width objects (for example, `ExchangeableWidth` objects) and properties that should be invisible. Once deployed, the rules become active when the `NoVVW` filter rule set is selected.

Example 2: You want to know which objects and properties are visible

```
% get all objects, and iterate them.
hDDObjects = dsdd('find',0,'regexp','.*');
for h=hDDObjects
   p = dsdd('GetAttribute',h,'NormalizedPath');
   if dsdd('IsVisible',h)
      fprintf('%s is visible.\n',p);
   else
      fprintf('%s is not visible.\n',p);
   end
   % the properties
   pn = dsdd('GetPropertyNames',h);
   for k=1:numel(pn)
      if dsdd('IsVisible',h,pn{k})
         fprintf('%s.%s is visible.\n',p,pn{k});
      else
         fprintf('%s.%s is not visible.\n',p,pn{k});
      end
   end
end
```

This dumps the visibility attribute of all objects and properties in DD0 into the MATLAB Command Window.

Example 3: You want to show all objects and properties

You want to have a set which specifies that all objects and properties are visible.

```
% create filter rule set "ShowAll",
% and make it the current filter rule set
dsdd('CreateFilterRuleSet','ShowAll');
dsdd('SetCurrentFilterRuleSet','ShowAll');
% dump all Data Model paths to file
dsdd('DumpDataModelPaths','DataModelPaths.log');
% read the paths line-by-line, and make it all visible
fid = fopen('DataModelPaths.log','r');
   while 1
      DataModelPath = fgetl(fid);
      if ~ischar(DataModelPath)
         break;
      end
      dsdd('SetFilterRule',DataModelPath,1);
end
fclose(fid);
delete('DataModelPaths.log');
```

```
% save the set to file - the filename is arbitrary
dsdd('WriteFilterRuleSet','file','ShowAll.xml');
```

The file `ShowAll.xml` now specifies that all objects and properties are visible. For example, the `PoolRef` property will appear in the list of unset properties of Variable objects even if the property could not be set.

Example 4: You want to deploy a filter rule set to be available in the DD Manager

You have a filter rule set `TemplatesAreOnlyForExperts.xml` and `ShowAll.xml`, and you want to deploy them so that the filter rule sets are available in each MATLAB session.

```
% Specify the path the files are copied to
tl_pref('set','DDFilterRuleDirectory','C:\myRules');
% Copy TemplatesAreOnlyForExperts.xml and ShowAll.xml to this
directory:
copyfile('TemplatesAreOnlyForExperts.xml','C:\myRules');
copyfile('ShowAll.xml,'C:\myRules\');
% The filter rule set is now deployed!
% For testing purposes, load the files with
bSuccess = dsdd('ReloadFilterRuleSets');
% or re-initialize the DD with
dsdd_free;
% The filter rules in the files become active immediately.
% The bSuccess return argument signals success or failure.
% On failure, you can check the messages
% in the DD Message List:
ds_error_register(dsdd('GetMessageList'));
ds_msgdlg('update');
```

Tip

You can also deploy the files in TargetLink's application data directory:

```
copyfile('TemplatesAreOnlyForExperts.xml,tl_env
('GetApplicationSettingsPath'));
copyfile('ShowAll.xml,tl_pref('get', 'ddfilterruledirectory'));
```

However, ensure that Use application data directory checkbox in the TargetLink Preferences Editor's Data Dictionary / Filter Rules pane is selected.

Note

Additional examples of filter rule sets can be downloaded from the TargetLink Product Support Center: http://www.dspace.com/tlpsc.

Related topics

Basics

HowTos

Basics on Managing DD Messages (DD Message List)

Introduction

The TargetLink Data Dictionary MATLAB API works with a message list which stores the messages produced by or relating to the TargetLink Data Dictionary.

Note

It is a good practice and strongly recommended to thoroughly check for errors after any DD MATLAB API functions. Refer to the Data Dictionary demos for examples of proper error handling.
When you start any of the demos for the first time (type `tl_demos` in the MATLAB Command Window), you are prompted to enter a root folder. This allows you to change the model without having administrator privileges. The folder is specified in the TargetLink Preferences Editor's General / Demo Models pane. By default, the Documents folder is selected:
`C:\Users\[User]\Documents\dSPACE\TargetLink\[Version]\Demos`

The MATLAB API provides you with a set of commands for working with the message list. You can access and remove messages on the message list. Additionally, you can suppress specific messages and limit the number of messages which are stored. For detailed information on the message list commands, refer to the following commands in the TargetLink Data Dictionary Reference:

Suppressing messages

- To prevent a specific message from being appended to the message list, use SuppressMessage.

Accessing the message list

- To get the most recent message from the message list, use GetLastMessage.
- To get a specific message from the message list, use GetMessage.
- To get the whole message list, use GetMessageList.
- To get the number of messages in the message list, use GetNumMessages.

Deleting messages

- To clear the complete message list, use ClearMessageList.
- To delete a single message from the message list, use DeleteMessage.

Limiting the number of messages

- By default, the number of messages on the message list is limited to 4096. To set the limit, use SetMaxNumMessages.
- To get the current limit, use GetMaxNumMessages.

Example

```
% cannot create Scalings in a VariableGroup
[hObj,err] = dsdd('Create','/Pool/Variables/SC_x','ObjectKind','Scaling');
if err,
    % get message struct
    msg = dsdd('GetLastMessage');
    fprintf('Command failed with error %d: %s\n',msg.number,msg.msg)
end
```

Utilities

To simplify error handling for dsdd commands, the **dsdd_check_msg** utility is provided, see the following example:

```
% cannot create Scalings in a VariableGroup
[hObj,err] = dsdd('Create','/Pool/Variables/SC_x','ObjectKind','Scaling');
if dsdd_check_msg(err), return; end
```

In the calling M file, you can check for errors in subfunctions with the command:

```
MySubfunction(arg,...)
if ds_error_check,
    disp('an error has been registered by MySubfunction')
end
```

You can use the following commands:

- **ds_error_check**
- ds_error_register (TargetLink API Reference)
- **ds_error_check**
- **ds_error_none**
- **ds_error_clear**
- **ds_error_set**

Related topics

References

ds_error_check (TargetLink API Reference)
ds_error_clear (TargetLink API Reference)
ds_error_none (TargetLink API Reference)
ds_error_set (TargetLink API Reference)

Basics on DD Environment Variables

Introduction

Environment variables are strings which you can define in the data dictionary. They exist only while the data dictionary is loaded in the memory and are not saved to the DD project file.

Additionally, you can define your own environment variables, for example, to provide path information. An environment variable which is included in a script is expanded with its current value at run time.

Working with environment variables

For detailed information on the commands for working with environment variables, refer to the following commands in the TargetLink Data Dictionary Reference:

- To get an environment variable, use GetEnv.
- To set an environment variable, use SetEnv.

When used in the path name of a DD object, environment variables must be enclosed by dollar signs.

Example

```
% get the name of the current DD project file
prjFile = dsdd('GetDDAttribute',0,'fileName');
% set environment variable BR
dsdd('SetEnv','BR','BR4711')
% use environment variable BR in a path name
cl = dsdd('GetClass','/Pool/Variables/$BR$/Kp');
```

Clearing TargetLink MEX DLLs from MATLAB Memory

Introduction

The TargetLink Data Dictionary MATLAB API is programmed as a MEX DLL, the dsdd MEX DLL. It is loaded when the first TargetLink model is opened.

When you no longer use the MATLAB API, you can unlock the dsdd MEX DLL and then remove it from memory.

- For detailed information on unlocking the dsdd MEX DLL, refer to the Unlock.

- The `dsdd_free` API function.

NOTICE

If the current DD project file has not been saved when you use this utility, data might be lost.

Related topics

References

dsdd_free (📖 TargetLink API Reference)

Working with Access Rights

Introduction

The TargetLink Data Dictionary provides two modes for access rights: administrator (admin access mode) and user (user access mode). For detailed information on access rights, refer to Basics on User Modes and Access Rights on page 19.

The MATLAB API provides commands that let you set a password for the administrator mode, switch between modes, and retrieve information on the current mode.

Commands for working with access rights

For detailed information on the commands for working with the access rights, refer to the following commands in the 📖 TargetLink Data Dictionary Reference:

- To set the password for administrator mode, use SetPassword.
- To set the user or administrator mode, use SetUser.
- To get information on the current user mode, use GetUser.

Example

```
bSuccess = dsdd('SetPassword',oldPW,newPW);
dsdd('SetUser','admin')
user = dsdd('GetUser');
```

Related topics

Basics

Basics on User Modes and Access Rights.. 19

Utilities for Easier Work with the TargetLink Data Dictionary

Introduction

The TargetLink Data Dictionary has some utilities that make your work much easier.

- To read the value of a DD Variable object (Value property) for the active variant, use `ddv`.
- To manage TargetLink Data Dictionary project files associated with TargetLink models, for example, to get the name of the DD project file that is associated with the current TargetLink model, use **`dsdd_manage_build`**.
- To manage an DD Application object, for example, to set or get the application name, use **`dsdd_manage_application`**.
- To validate DD objects, use `dsdd_validate`.
- To manage a DD Build object, use `dsdd_manage_build`.
- To compare two different DD project files, use `dsddman('Compare', propertyName, propertyValue, ...)dsdd_compare`.
- To get the Simulink block or Stateflow object associated with a DD object, use `dsdd_get_block_path`.

Related topics

References

ddv (📖 TargetLink API Reference)
dsdd_get_block_path (📖 TargetLink API Reference)
dsdd_manage_application (📖 TargetLink API Reference)
dsdd_manage_build (📖 TargetLink API Reference)
dsdd_validate (📖 TargetLink API Reference)
dsddman('Compare', propertyName, propertyValue, ...) (📖 TargetLink API Reference)

Adding Custom Functionality to the Data Dictionary Manager

Introduction

The TargetLink Data Dictionary lets you add custom functionality, i.e., user-defined MATLAB functions (for example, M files) to the Data Dictionary Manager.

Where to go from here

Information in this section

Basics on Adding Custom Functionality to the Data Dictionary Manager

Introduction

The TargetLink Data Dictionary lets you add custom functionality to the TargetLink Data Dictionary Manager:

- User-specific menu extensions
- Custom output views

- Custom messages
- Custom command calls

Note

This feature is not available in stand-alone mode, i.e., it is available only when the TargetLink Data Dictionary Manager is started via TargetLink or the MATLAB Command Window.

Menu commands

You can extend menus with user-defined menu commands that call user-defined MATLAB functions (M files).

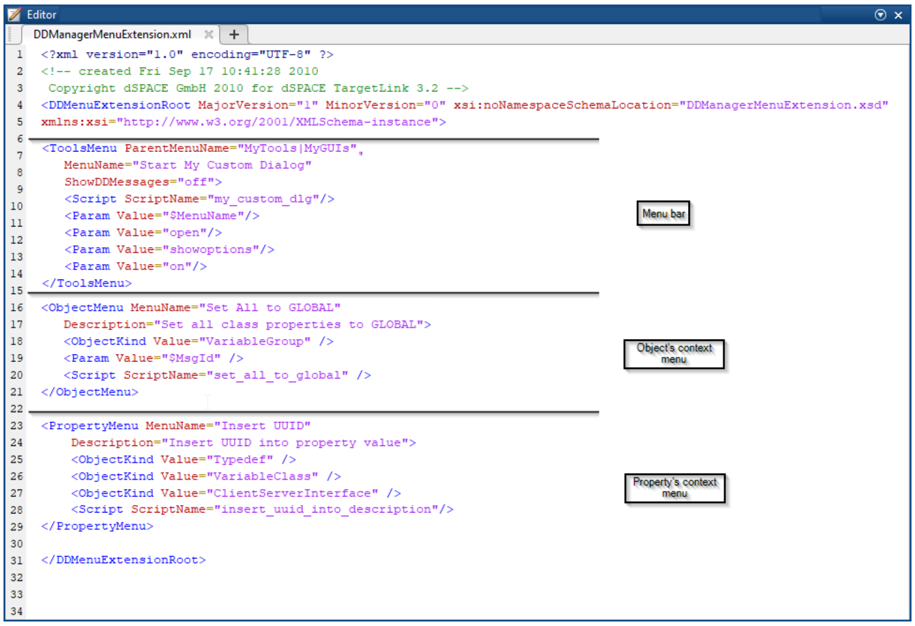
You can specify menu commands in

- The menu bar
- The context menus of objects in the Data Dictionary Navigator
- The context menus of properties in the Property Value List

Specifying user-specific menu extensions

To view and alter the menu extension set select Edit Menu Extension Specification from the Extras menu which opens the definition file `DDManagerMenuExtension.xml` in the currently defined code editor.

The default is MATLAB's built-in editor. You can change it via the Preferences Editor. For details, refer to Basics on Using the Preferences Editor (📖 TargetLink Customization and Optimization Guide).



```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- created Fri Sep 17 10:41:28 2010
 Copyright dSPACE GmbH 2010 for dSPACE TargetLink 3.2 -->
<DDMenuExtensionRoot MajorVersion="1" MinorVersion="0" xsi:noNamespaceSchemaLocation="DDManagerMenuExtension.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<ToolsMenu ParentMenuName="MyTools|MyGUIs",
   MenuName="Start My Custom Dialog"
   ShowDDMessages="off">
   <Script ScriptName="my_custom_dlg"/>
   <Param Value="$MenuName"/>
   <Param Value="open"/>
   <Param Value="showoptions"/>
   <Param Value="on"/>
</ToolsMenu>

<ObjectMenu MenuName="Set All to GLOBAL"
   Description="Set all class properties to GLOBAL">
   <ObjectKind Value="VariableGroup" />
   <Param Value="$MsgId" />
   <Script ScriptName="set_all_to_global" />
</ObjectMenu>

<PropertyMenu MenuName="Insert UUID"
    Description="Insert UUID into property value">
    <ObjectKind Value="Typedef" />
    <ObjectKind Value="VariableClass" />
    <ObjectKind Value="ClientServerInterface" />
    <Script ScriptName="insert_uuid_into_description"/>
</PropertyMenu>

</DDMenuExtensionRoot>
```

Once the file is modified, you can reload it by clicking Reload Menu Extension Specification in the Extras menu.

Messages about success or failure are displayed in the **Message Browser**. This lets you find and correct specification errors.

You can also specify menu extensions in separate XML files. You have to use the same structure and syntax as in `DDManagerMenuExtension.xml` for this. The additional files have to be stored on the same path as the `DDManagerMenuExtension.xml` file. The current path is displayed in the title bar of the built-in editor in MATLAB.

Custom messages and custom output views

If you want to display feedback from your own tools, you can issue custom messages in either the **Message Browser** or a custom output view, i.e., a separate pane that you can create in the Data Dictionary Manager. For further information, refer to How to Create Custom Output Views on page 146 and How to Display Custom Messages on page 147.

Related topics

HowTos

How to Add User-Specific Menu Extensions to Objects' Context Menus in the Data Dictionary Navigator........ 140
How to Add User-Specific Menu Extensions to Properties' Context Menus in the Property Value List........ 143
How to Add User-Specific Menu Extensions to the Menu Bar........ 137
How to Create Custom Output Views........ 146
How to Display Custom Messages........ 147

References

Edit Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)
Reload Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)

How to Add User-Specific Menu Extensions to the Menu Bar

Objective

To add user-specific menu extensions to the Data Dictionary Manager's menu bar by extending it with menu commands that call MATLAB functions with an arbitrary number of constant string parameters.

Precondition

You have to provide the MATLAB function that you want to call via the menu bar, for example, in an M file.

The syntax of the MATLAB command is:

```
<functionName>[(<param_1>,<param_2>, ... , <param_n>)]
```

Method

To add user-specific menu extensions to the menu bar

1 From the Extras menu of the Data Dictionary Manager, select Edit Menu Extension Specification.

The current Code Editor (by default MATLAB's built-in editor) opens the `DDManagerMenuExtension.xml` file that contains the specifications of the menu extensions.

2 Enter the specification for the menu command that you want to add to the menu bar, using the following syntax:

```
<ToolsMenu ParentMenuName=<ParentMenuName>
    MenuName=<MenuName>
    Description=<Description>
    ShowDDMessages=<showDDMessages>
    Separator=<separator>>
   <Script ScriptName=<functionName>/>
   <Param Value=<value>/>
</ToolsMenu>
```

For more information on the syntax, refer to Edit Menu Extension Specification (TargetLink Data Dictionary Manager Reference)

3 Save the file.

4 For the changes to take effect, click Reload Menu Extension Specification in the Extras menu to reload the menu extension specification to the Data Dictionary Manager.

Result

You have added a Tools menu command to the Data Dictionary Manager's menu bar that calls a MATLAB function.

Displaying messages of the custom functionality

If you want to display feedback from your functions, you can issue custom messages in either the Message Browser or a custom output view, i.e., a separate pane that you can create in the Data Dictionary Manager. For further information, refer to How to Create Custom Output Views on page 146 and How to Display Custom Messages on page 147.

Examples

Start My Custom Dialog For this example, you have to define the following function in MATLAB:

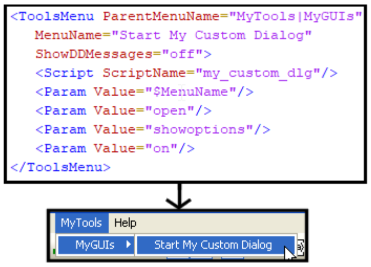
```
function my_custom_dlg(menuName, cmdName, paramName, paramValue)

        msg = sprintf('My custom dialog %s was called with cmdName=%s, %s=%s', menuName, cmdName, paramName, paramValue);
        msgbox(msg, 'My custom dialog');

    end
```

Save the function under the file name my_custom_dlg.m.

In the Data Dictionary Manager, go toExtras-Edit Menu Extension Specification, and enter the following XML code:

```
<ToolsMenu  ParentMenuName="MyTools|MyGUIs"
   MenuName="Start My Custom Dialog"
   Description="My special custom dialog."
   ShowDDMessages="off">
   <Script ScriptName="my_custom_dlg"/>
   <Param Value="$MenuName"/>
   <Param Value="open"/>
   <Param Value="showoptions"/>
   <Param Value="on"/>
</ToolsMenu>
```

This piece of XML code specifies that a menu command called Start My Custom Dialog is appended to the MyTools-MyGUIs menu.

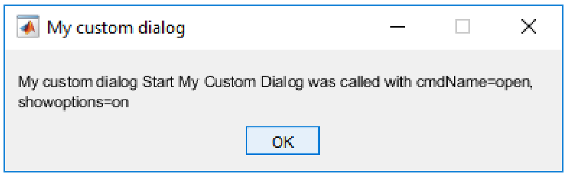


```
<ToolsMenu ParentMenuName="MyTools|MyGUIs"
   MenuName="Start My Custom Dialog"
   ShowDDMessages="off">
   <Script ScriptName="my_custom_dlg"/>
   <Param Value="$MenuName"/>
   <Param Value="open"/>
   <Param Value="showoptions"/>
   <Param Value="on"/>
</ToolsMenu>
```

When you click this menu command, the `my_custom_dlg` MATLAB function is called using the syntax:

```
my_custom_dlg('Start My Custom Dialog','open','showoptions','on')
```

The following message is displayed:



My custom dialog

My custom dialog Start My Custom Dialog was called with cmdName=open, showoptions=on

Click OK to close this dialog.

Related topics

Basics

HowTos

References

Edit Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)
Reload Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)

How to Add User-Specific Menu Extensions to Objects' Context Menus in the Data Dictionary Navigator

Objective

To add user-specific menu extensions to an object's context menu by adding object-kind-specific menu commands that call MATLAB functions.

Note

When multiple DD objects are selected, object context menus are extended only if the DD objects are of the same object kind.

Precondition

You have to provide the MATLAB function that you want to call via the context menu of an object in the **Data Dictionary Navigator**, for example, in an M file.

The syntax of the MATLAB command is:

```
<functionName>(<hDDObject>)
```

`<hDDObject>` is the DD object handle that specifies the selected DD object.

Method

To add user-specific menu extensions to an object's context menu in the Data Dictionary Navigator

1 In the Data Dictionary Manager, select Edit Menu Extension Specification from the Extras menu.

The current Code Editor (by default MATLAB's built-in editor) opens the `DDManagerMenuExtension.xml` file, which contains the specifications of the menu extensions.

2 Enter the specification for the menu command that you want to add to an object's context menu in the Data Dictionary Navigator, using the following syntax:

```
<ObjectMenu ParentMenuName=<ParentMenuName>
    MenuName=<MenuName>
    Description=<Description>
    ShowDDMessages=<showDDMessages>
    Separator=<separator>>
    <ObjectKind Value=<objectKind>/>
    <Script ScriptName=<functionName>/>
</ObjectMenu>
```

For more information on the syntax, refer to Edit Menu Extension Specification (TargetLink Data Dictionary Manager Reference).

3 Save the file.

4 For the changes to take effect, click Reload Menu Extension Specification in the Extras menu to reload the menu extension specification to the Data Dictionary Manager.

Result

You added a menu command that calls a MATLAB function to an object's context menu in the Data Dictionary Navigator. This context menu command is added for each DD object whose object kind is one of those specified in the `DDManagerMenuExtension.xml`.

Displaying messages of the custom functionality

If you want to display feedback from your functions, you can issue custom messages in either the Message Browser or a custom output view, i.e., a separate pane that you can create in the Data Dictionary Manager. For further information, refer to How to Create Custom Output Views on page 146 and How to Display Custom Messages on page 147.

Example

For this example, you have to define the following function in MATLAB:

```
function set_all_to_global(hDDVarGroup, msgId)

    hChildren = dsdd('GetChildren', hDDVarGroup);
    msg = ['Following objects are set to global:' newline];
    for i=1:length(hChildren)
        attr = dsdd('GetAttributes', hChildren(i));
        if strcmpi(attr.objectKind, 'variable')
            dsdd('SetClass', hChildren(i), 'GLOBAL');
            msg = [msg attr.name newline];

        end
    end

    dsddman('AddMessage', 'Title', 'Set All To Global', ...
            'Message', msg, 'Reference', hDDVarGroup, ...
            'RefKind', 'ddobject', 'ID', msgId, ...
            'Type', 'Info');

    end
```

Save the function under the file name set_all_to_global.m.

```
<ObjectMenu MenuName="Set All to GLOBAL"
       Description="Set all class properties to GLOBAL">
       <ObjectKind Value="VariableGroup" />
       <Param Value="$MsgId" />
       <Script ScriptName="set_all_to_global" />
    </ObjectMenu>
```

This piece of XML code specifies that a menu called Set All to GLOBAL is appended to the context menu of all VariableGroup objects. You can use the *pipe* symbol (|) to create nested menus.

When you click this menu, the `set_all_to_global` MATLAB function is called using the following syntax:

```
set_all_to_global(<hDDVarGroup>, <MessageID>)
```

`<hDDVarGroup>` is the DD handle of the selected VariableGroup object and `<MessageID>` is the message identifier of the Data Dictionary Manager parent message. If more than one object is selected, the object handles are passed to the MATLAB function as a vector. DD messages are displayed in the Message Browser after the function is executed.

Related topics

Basics

Basics on Adding Custom Functionality to the Data Dictionary Manager................................ 135

HowTos

How to Add User-Specific Menu Extensions to Properties' Context Menus in the Property Value List.. 143
How to Add User-Specific Menu Extensions to the Menu Bar.. 137
How to Create Custom Output Views.. 146
How to Display Custom Messages... 147

References

Edit Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)
Reload Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)

How to Add User-Specific Menu Extensions to Properties' Context Menus in the Property Value List

Objective

To add user-specific menu extensions to a property's context menu by adding property-specific menu commands that call MATLAB functions.

Precondition

You have to provide the MATLAB function that you want to call via the context menu of a property in the Property Value List, for example, in an M file.

The syntax of the MATLAB command is:

```
<functionName>(<hDDObject>,<propertyName>)
```

`<hDDObject>` is the DD object handle that specifies the DD object which owns the property. `<propertyName>` is the name of the property.

Method

To add user-specific menu extensions to a property's context menu in the Property Value List

1 In the Data Dictionary Manager, select Edit Menu Extension Specification on the Extras menu.

The current code editor (by default built-in MATLAB Editor) opens the `DDManagerMenuExtension.xml` file, which contains the specifications for the menu extensions.

2 Enter the specification for the menu command that you want to add to a property's context menu in the **Property Value List**, using the following syntax:

```
<PropertyMenu ParentMenuName=<ParentMenuName>
    MenuName=<MenuName>
    Description=<Description>
    ShowDDMessages=<showDDMessages>
    Separator=<separator>>
    <ObjectKind Value=<objectKind>/>
    <PropertyName Value=<propertyName>/>
    <Script ScriptName=<functionName>/>
</PropertyMenu>
```

For more information on the syntax, refer to Edit Menu Extension Specification (TargetLink Data Dictionary Manager Reference)

3 Save the file.

4 For the changes to take effect, click **Reload Menu Extension Specification** in the **Extras** menu to reload the menu extension specification to the Data Dictionary Manager.

Note

If you specify more than one **object kind** and more than one **property name** the context menu will be available for each property of a DD object whose object kind is in the list of specified object kinds and whose name is in the list of specified property names.

Result

You have added a context menu command that calls a MATLAB function for a property in the **Property Value List**.

Displaying messages of the custom functionality

If you want to display feedback from your functions, you can issue custom messages in either the Message Browser or a custom output view, i.e., a separate pane that you can create in the Data Dictionary Manager. For further information, refer to How to Create Custom Output Views on page 146 and How to Display Custom Messages on page 147.

Example

For this example, you have to define the following function in MATLAB:

```
function insert_uuid_into_description(hDDObject, strPropName)

    uuid = char(java.util.UUID.randomUUID);
    dsdd('Set', hDDObject, strPropName, uuid);

    end
```

Save the function under the file name insert_uuid_into_description.m.

```
<PropertyMenu MenuName="Insert UUID"
    Description="Insert UUID into property value">
    <ObjectKind Value="Typedef" />
    <ObjectKind Value="VariableClass" />
    <ObjectKind Value="ClientServerInterface" />
    <Script ScriptName="insert_uuid_into_description"/>
</PropertyMenu>
```

This piece of XML code specifies that a menu command called Insert UUID is appended to the context menu of all Description and Uuid properties that belong to Typedef, VariableClass or ClientServerInterface objects.

When you click this menu, the `insert_uuid_into_description` MATLAB function is called using the following syntax:

```
insert_uuid_into_description(<hDDObject>,<PropertyName>)
```

`<hDDObject>` is the DD handle of the selected DD object and `<Propertyname>` is the name of the property on which the context menu was called. DD messages are displayed in the Message Browser after the function is executed.

Related topics

Basics

Basics on Adding Custom Functionality to the Data Dictionary Manager................................ 135

HowTos

How to Add User-Specific Menu Extensions to Objects' Context Menus in the Data Dictionary Navigator.. 140
How to Add User-Specific Menu Extensions to the Menu Bar.. 137
How to Create Custom Output Views.. 146
How to Display Custom Messages... 147

References

Edit Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)
Reload Menu Extension Specification (📖 TargetLink Data Dictionary Manager Reference)

How to Create Custom Output Views

Objective

To create a custom output view that displays your own messages in a separate pane of the Data Dictionary Manager.

Precondition

The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.

Method

To create a custom output view

1 In the MATLAB Command Window, type

```
dsddman('DemandCustomOutputView','Name','<name>')
```

Result

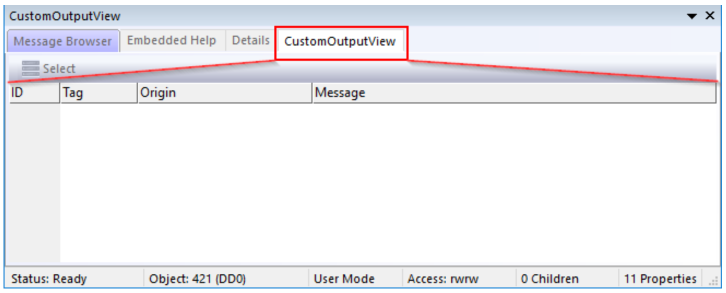
A custom output view with the specified name opens. Additionally, a temporary menu command with the specified name is created in the View — Show Output View menu to let you show and hide the custom output view.

You can use it to issue custom messages. For further information, refer to How to Display Custom Messages on page 147.

Examples

```
dsddman('DemandCustomOutputView','Name','CustomOutputView')
```

creates a custom output view called CustomOutputView.



Related topics

Basics

HowTos

How to Display Custom Messages

Objective

To issue custom messages in either the Message Browser or a custom output view that display feedback from your own tools in the Data Dictionary Manager.

Preconditions

The Data Dictionary Manager is open. Refer to How to Start the Data Dictionary Manager on page 37.

Method 1

To issue a custom message in the Message Browser

1 In the MATLAB Command Window, type

```
dsddman('AddMessage', ...
'Title','<title>', ...
'Message','<message>', ...
'Reference','<reference>','Refkind','<'refkind>', ...
'ID','<id>')
```

Method 2

To issue a custom message in a custom output view

1 In the MATLAB Command Window, type

```
dsddman('AddCustomMessage', ...
'Name','<name>', ...
'Message','<message>', ...
'Reference','<reference>','Refkind','<refkind>', ...
'Tag','<tag>')
```

Attributes

The following table shows the attributes for issuing a message in the Message Browser or a custom output view:

Name	Description	
`Name`	Name of the previously created custom output view	
`Message`	Message text (must not be empty)	
`Title`	Message title	
`Reference`	Object associated with the message (must be used with property `Refkind`)	
`RefKind`	Type of the object specified by property `Reference`	
	`slblock`	Simulink block
	`sfobject`	Stateflow object
	`ddobject`	DD object
	`file`	File
	`mxarray`	MATLAB array
`Id`	ID of the parent message for the displayed custom message[1)] If -1, the custom message is added to the root of the message hierarchy as the last element.	
`Tag`	Tag (can be used to introduce message categories, for example, Info)	

1) For a MATLAB function called via a menu extension, the Data Dictionary Manager issues a message and creates an ID. If you use `<Param Value="$MsgId"/>`, child messages can be added to the parent message and are shown in the Message Browser. For more information, refer to Basics on Adding Custom Functionality to the Data Dictionary Manager on page 135.

Result

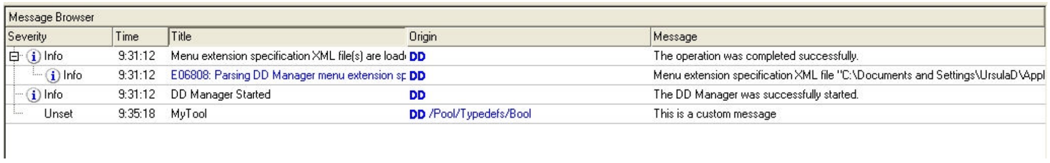
You have issued a custom message.

Examples

Custom message in Message Browser

```
dsddman('AddMessage',...
'Title','MyTool',...
'Message','This is a custom message',...
'reference','/Pool/Typedefs/Bool','Refkind','ddobject',...
'ID',-1)
```

issues the following message in the Message Browser.



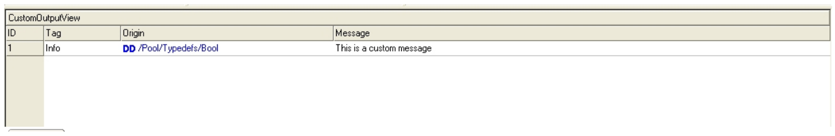
Message Browser

Severity	Time	Title	Origin	Message
Info	9:31:12	Menu extension specification XML file(s) are load	DD	The operation was completed successfully.
Info	9:31:12	E06808: Parsing DD Manager menu extension sp	DD	Menu extension specification XML file "C:\Documents and Settings\UrsulaD\Appl
Info	9:31:12	DD Manager Started	DD	The DD Manager was successfully started.
Unset	9:35:18	MyTool	DD /Pool/Typedefs/Bool	This is a custom message

Custom message in custom output view

```
dsddman('AddCustomMessage',...
'Name','CustomOutputView',...
'Message','This is a custom message',...
'reference','/Pool/Typedefs/Bool','Refkind','ddobject',...
'Tag','Info');
```

issues the following message in the CustomOutputView output view created in the example in How to Create Custom Output Views on page 146.



CustomOutputView

ID	Tag	Origin	Message
1	Info	DD /Pool/Typedefs/Bool	This is a custom message

Related topics

Basics

HowTos

How to Specify Custom Command Calls

Objective

To call your own commands from within the Data Dictionary by specifying CustomCommand objects. You can also specify additional arguments.

Custom commands

Custom commands can be external programs, your own M scripts or other preparatory scripts or functions you want to call. This can be especially useful for administrators who want to customize specific work environments for team members by providing DD file templates containing custom commands.

Sequential command calls (grouping)

You can call multiple custom commands by using CustomCommandGroup objects. For a command call sequence, place all the relevant CustomCommand objects in the group object.

Windows Command Prompt or MATLAB environment

You can customize command calls for a Windows Command Prompt (e.g., to call external programs) or for the MATLAB environment (e.g., to call M scripts).

Using macros with custom properties

You can use macros with corresponding custom properties. If you want to change specific or multiple arguments of commands, you can use this mechanism to easily change arguments. In the example below, `$(FileName)` is a macro that needs a corresponding custom property with the name of the file to be opened.

Note

Do not confuse this macros with the name macros described in Basics on Using Name Macros (📖 TargetLink Customization and Optimization Guide).

Demo model

You can observe a demo model to learn more about custom command calls, custom properties and different command call environments. Refer to DD_CUSTOM_COMMAND (📖 TargetLink Demo Models).

Method

To specify a custom command call

1 From the context menu of the `Pool` area in the Data Dictionary Manager, select Create CustomCommands to create a group for the custom command objects.

2 From the context menu of the CustomCommands group object, select Create CustomCommand.

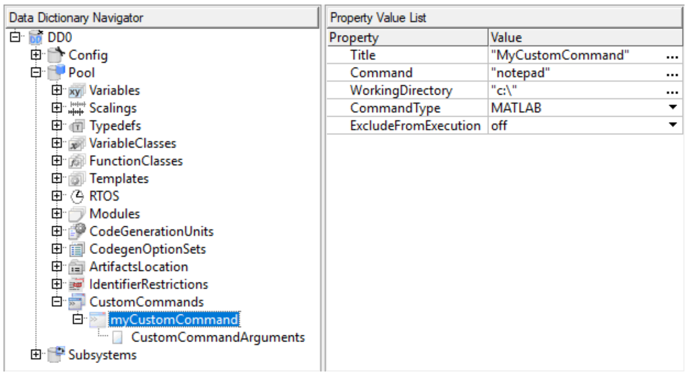
3 You can specify the command name and its working directory. For example, if you want to open a simple `ReadMe.txt` file in Windows Notepad, click the `Name` property of the CustomCommand object and enter `Notepad`.

4 Specify your working directory.

5 Set the `CommandType` property to `DOS`. This indicates that the command is to be called in a Windows Command Prompt.

6 From the context menu of your CustomCommand object, select Create CustomCommandArguments.

A child object is created.



Data Dictionary Navigator

```
DD0
  Config
  Pool
    Variables
    Scalings
    Typedefs
    VariableClasses
    FunctionClasses
    Templates
    RTOS
    Modules
    CodeGenerationUnits
    CodegenOptionSets
    ArtifactsLocation
    IdentifierRestrictions
    CustomCommands
      myCustomCommand
        CustomCommandArguments
  Subsystems
```

Property Value List

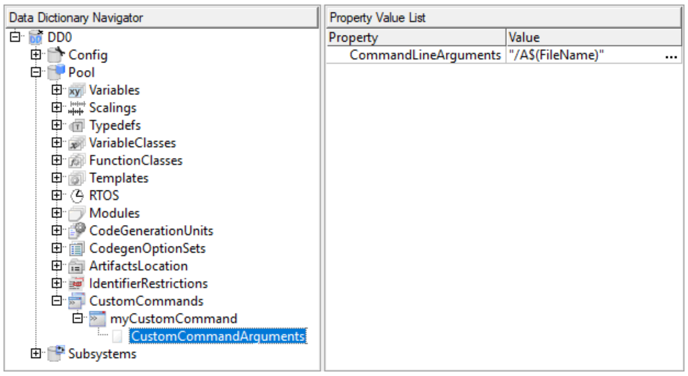
Property	Value
Title	"MyCustomCommand"
Command	"notepad"
WorkingDirectory	"c:\"
CommandType	MATLAB
ExcludeFromExecution	off

7 You must now specify the file name (and optionally the working directory) in the CustomCommandArguments object. Select the `CommandLineArguments` property and enter `/A $(FileName)`. `/A` is a Notepad argument to open the file as ANSI.`$(FileName)` is a custom command macro that needs a corresponding custom property with the name of the file to be opened.

8 From the context menu of the CustomCommandArguments object, select Create custom property.

9 From the context menu of the CustomCommands group object, select Add custom property - String.

10 Rename the property to `FileName` as defined above and enter `ReadMe.txt` as shown below.



Data Dictionary Navigator

```
DD0
  Config
  Pool
    Variables
    Scalings
    Typedefs
    VariableClasses
    FunctionClasses
    Templates
    RTOS
    Modules
    CodeGenerationUnits
    CodegenOptionSets
    ArtifactsLocation
    IdentifierRestrictions
    CustomCommands
      myCustomCommand
        CustomCommandArguments
  Subsystems
```

Property Value List

Property	Value
CommandLineArguments	"/A$(FileName)"

11 From the context menu Custom Command object, select Run custom command.

`/A` and `ReadMe.txt` are passed as arguments to Notepad. During command execution, Notepad starts and opens the `ReadMe.txt` file in ANSI.

Note

Using macros is optional. You can also enter the arguments for the command directly in the value field of the `CommandLineArguments` property. However, they become useful if you have lots of arguments and want to create different custom command calls for different use cases.

Result

You specified a custom command call.

Related API function

You can also use the MATLAB API function. Refer to tlExecuteDDCustomCommand (📖 TargetLink API Reference).

Related topics

Basics

DD_CUSTOM_COMMAND (📖 TargetLink Demo Models)

Glossary

Introduction	The glossary briefly explains the most important expressions and naming conventions used in the TargetLink documentation.

Where to go from here

Information in this section

Numerics

1-D look-up table A look-up table that maps one input value (x) to one output value (y).

2-D look-up table A look-up table that maps two input values (x,y) to one output value (z).

A

Abstract interface An interface that allows you to map a project-specific, physical specification of an interface (made in the TargetLink Data Dictionary) to a logical interface of a modular unit. If the physical interface changes, you do not have to change the Simulink subsystem or the partial DD file and therefore neither the generated code of the modular unit.

Access function (AF) A C function or function-like preprocessor macro that encapsulates the access to an interface variable.

See also read/write access function and variable access function.

Acknowledgment Notification from the RTE that a data element or an event message have been transmitted.

Activating RTE event An RTE event that can trigger one or more runnables.

See also activation reason.

Activation reason The activating RTE event that actually triggered the runnable.

Activation reasons can group several RTE events.

Active page pointer A pointer to a data page. The page referenced by the pointer is the active page whose values can be changed with a calibration tool.

Adaptive AUTOSAR Short name for the AUTOSAR *Adaptive Platform* standard. It is based on a service-oriented architecture that aims at on-demand software updates and high-end functionalities. It complements Classic AUTOSAR.

Adaptive AUTOSAR behavior code Code that is generated for model elements in Adaptive AUTOSAR Function subsystems or Method Behavior subsystems. This code represents the behavior of the model and is part of an adaptive application. Must be integrated in conjunction with ARA adapter code.

Adaptive AUTOSAR Function A TargetLink term that describes a C++ function representing a partial functionality of an adaptive application. This function can be called in the C++ code of an adaptive application. From a higher-level perspective, Adaptive AUTOSAR functions are analogous to runnables in Classic AUTOSAR.

Adaptive AUTOSAR Function subsystem An atomic subsystem used to generate code for an Adaptive AUTOSAR Function. It contains a Function block whose AUTOSAR mode is set to `Adaptive` and whose Role is set to `Adaptive AUTOSAR Function`.

ANSI C Refers to C89, the C language standard ANSI X3.159-1989.

Application area An optional DD object that is a child object of the DD root object. Each Application object defines how an ECU program is built from the generated subsystems. It also contains some experiment data, for example, a list of variables to be logged during simulations and results of code coverage tests.

Build objects are children of Application objects. They contain all the information about the binary programs built for a certain target platform, for example, the symbol table for address determination.

Application data type Abstract type for defining types from the application point of view. It allows you to specify physical data such as measurement data. Application data types do not consider implementation details such as bit-size or endianness.

Application data type (ADT) According to AUTOSAR, application data types are used to define types at the application level of abstraction. From the application point of view, this affects physical data and its numerical representation. Accordingly, application data types have physical semantics but do not consider implementation details such as bit width or endianness.

Application data types can be constrained to change the resolution of the physical data's representation or define a range that is to be considered.

See also implementation data type (IDT).

Application layer The topmost layer of the ECU software. The application layer holds the functionality of the ECU software and consists of atomic software components (atomic SWCs).

ARA adapter code Adapter code that connects Adaptive AUTOSAR behavior code with the Adaptive AUTOSAR API or other parts of an adaptive application.

Array-of-struct variable An array-of-struct variable is a structure that either is non-scalar itself or that contains at least one non-scalar substructure at any nesting depth. The use of array-of-struct variables is linked to arrays of buses in the model.

Artifact A file generated by TargetLink:

- Code coverage report files
- Code generation report files
- Metadata files
- Model-linked code view files
- Production code files
- Simulation application object files
- Simulation frame code files
- Stub code files

Artifact location A folder in the file system that contains an artifact. This location is specified relatively to a project folder.

ASAP2 File Generator A TargetLink tool that generates ASAP2 files for the parameters and signals of a Simulink model as specified by the corresponding TargetLink settings and generated in the production code.

ASCII In production code, strings are usually encoded according to the ASCII standard. The ASCII standard is limited to a set of 127 characters implemented by a single byte. This is not sufficient to display special characters of different languages. Therefore, use another character encoding, such as UTF-8, if required.

Asynchronous operation call subsystem A subsystem used when modeling *asynchronous* client-server communication. It is used to generate the call of the `Rte_Call` API function and for simulation purposes.

See also operation result provider subsystem.

Asynchronous server call returns event An RTE event that specifies whether to start or continue the execution of a runnable after the execution of a server runnable is finished.

Atomic software component (atomic SWC) The smallest element that can be defined in the application layer. An atomic SWC describes a single functionality and contains the corresponding algorithm. An atomic SWC communicates with the outside only via the interfaces at the SWC's ports. An atomic SWC is defined by an internal behavior and an implementation.

Atomic software component instance An atomic software component (atomic SWC) that is actually used in a controller model.

AUTOSAR Abbreviation of *AUT*omotive *O*pen *S*ystem *AR*chitecture. The AUTOSAR partnership is an alliance in which the majority of OEMs, suppliers, tool providers, and semiconductor companies work together to develop and establish a de-facto open industry-standard for automotive electric/electronics (E/E) architecture and to manage the growing E/E complexity.

AUTOSAR import/export Exchanging standardized software component descriptions between AUTOSAR tools.

AUTOSAR subsystem An atomic subsystem that contains a Function block whose AUTOSAR mode property is set to `Classic`. See also operation subsystem, operation call with runnable implementation subsystem, and runnable subsystem.

AUTOSAR tool Generic term for the following tools that are involved in the ECU network software development process according to AUTOSAR:

- Behavior modeling tool
- System-level tool
- ECU-centric tool

TargetLink acts as a behavior modeling tool in the ECU network software development process according to AUTOSAR.

Autoscaling Scaling is performed by the Autoscaling tool, which calculates worst-case ranges and scaling parameters for the output, state and parameter variables of TargetLink blocks. The Autoscaling tool uses either worst-case ranges or simulated ranges as the basis for scaling. The upper and lower worst-case range limits can be calculated by the tool itself. The Autoscaling tool always focuses on a subsystem, and optionally on its underlying subsystems.

B

Basic software The generic term for the following software modules:

- System services (including the operating system (OS) and the ECU State Manager)
- Memory services (including the NVRAM manager)
- Communication services
- I/O hardware abstraction
- Complex device drivers

Together with the RTE, the basic software is the platform for the application layer.

Batch mode The mode for batch processing. If this mode is activated, TargetLink does not open any dialogs. Refer to How to Set TargetLink to Batch Mode (TargetLink Orientation and Overview Guide).

Behavior model A model that contains the control algorithm for a controller (function prototyping system) or the algorithm of the controlled system (hardware-in-the-loop system). Can be connected in ConfigurationDesk via model ports to build a real-time application (RTA). The RTA can be executed on real-time hardware that is supported by ConfigurationDesk.

Block properties Properties belonging to a TargetLink block. Depending on the kind of the property, you can specify them at the block and/or in the Data Dictionary. Examples of block properties are:

- Simulink properties (at a masked Simulink block)
- Logging options or saturation flags (at a TargetLink block)
- Data types or variable classes (referenced from the DD)
- Variable values (specified at the block or referenced from the DD)

Bus A bus consists of subordinate bus elements. A bus element can be a bus itself.

Bus element A bus element is a part of a bus and can be a bus itself.

Bus port block Bus Inport, Bus Outport are bus port blocks. They are similar to the TargetLink Input and Output blocks. They are virtual, and they let you configure the input and output signals at the boundaries of a TargetLink subsystem and at the boundaries of subsystems that you want to generate a function for.

Bus signal Buses combine multiple signals, possibly of different types. Buses can also contain other buses. They are then called nested buses.

Bus-capable block A block that can process bus signals. Like bus port blocks, they allow you to assign a type definition and, therefore, a variable class to all the bus elements at once. The following blocks are bus-capable:

- Constant
- Custom Code (type II) block
- Data Store Memory, Data Store Read, and Data Store Write

- Delay
- Function Caller
- ArgIn, ArgOut
- Merge
- Multiport Switch (Data Input port)
- Probe
- Sink
- Signal Conversion
- Switch (Data Input port)
- Unit Delay
- Stateflow Data
- MATLAB Function Data

C

Calibratable variable Variable whose value can be changed with a calibration tool during run time.

Calibration Changing the calibration parameter values of ECUs.

Calibration parameter Any ECU variable type that can be calibrated. The term *calibration parameter* is independent of the variable type's dimension.

Calprm Defined in a calprm interface. Calprms represent calibration parameters that are accessible via a measurement and calibration system.

Calprm interface An interface that is provided or required by a software component (SWC) via a port (AUTOSAR).

Calprm software component A special software component (SWC) that provides calprms. Calprm software components have no internal behavior.

Canonical In the DD, array-of-struct variables are specified canonically. Canonical means that you specify one array element as a representative for all array elements.

Catalog file (CTLG) A description of the content of an SWC container. It contains file references and file category information, such as source code files (`C` and `H`), object code files (such as `O` or `OBJ`), variable description files (`A2L`), or AUTOSAR files (`ARXML`).

Characteristic table (Classic AUTOSAR) A look-up table as described by Classic AUTOSAR whose values are measurable or calibratable. See also compound primitive data type

Classic AUTOSAR Short name for the AUTOSAR *Classic Platform* standard that complements Adaptive AUTOSAR.

Classic initialization mode The initialization mode used when the Simulink diagnostics parameter Underspecified initialization detection is set to `Classic`.

See also simplified initialization mode.

Client port A require port in client-server communication as described by Classic AUTOSAR. In the Data Dictionary, client ports are represented as DD ClientPort objects.

Client-server interface An interface that describes the operations that are provided or required by a software component (SWC) via a port (AUTOSAR).

Code generation mode One of three mutually exclusive options for generating TargetLink standard production code, AUTOSAR-compliant production code or RTOS-compliant (multirate RTOS/OSEK) production code.

Code generation unit (CGU) The smallest unit for which you can generate code. These are:

- TargetLink subsystems
- Subsystems configured for incremental code generation
- Referenced models
- DD CodeGenerationUnit objects

Code output style definition file To customize code formatting, you can modify a code output style definition file (XML file). By modifying this file, you can change the representation of comments and statements in the code output.

Code output style sheets To customize code formatting, you can modify code output style sheets (XSL files).

Code section A section of generated code that defines and executes a specific task.

Code size Amount of memory that an application requires specified in RAM and ROM after compilation with the target cross-compiler. This value helps to determine whether the application generated from the code files fits in the ECU memory.

Code variant Code variants lead to source code that is generated differently depending on which variant is selected (i.e., varianted at code generation time). For example, if the Type property of a variable has the two variants Int16 and Float32, you can generate either source code for a fixed-point ECU with one variant, or floating-point code with the other.

Compatibility mode The default operation mode of RTE generators. The object code of an SWC that was compiled against an application header generated in compatibility mode can be linked against an RTE generated in compatibility mode (possibly by a different RTE generator). This is due to using standardized data structures in the generated RTE code.

See also vendor mode.

Compiler inlining The process of replacing a function call with the code of the function body during compilation by the C compiler via inline expansion.

This reduces the function call overhead and enables further optimizations at the potential cost of larger code size.

Composition A structuring element in the application layer. A composition consists of software components and their interconnections via ports.

Compound primitive data type A primitive application data type (ADT) as defined by Classic AUTOSAR whose category is one of the following:

- COM_AXIS
- CUBOID
- CUBE_4
- CUBE_5
- CURVE
- MAP
- RES_AXIS
- VAL_BLK
- STRING

Compute-through-overflow (CTO) Calculation method for additions and subtraction where overflows are allowed in intermediate results without falsifying the final result.

Concern A concept in component-based development. It describes the idea that components separate their concerns. Accordingly, they must be developed in such a way that they provide the required functionality, are flexible and easy to maintain, and can be assembled, reused, or replaced by newer, functionally equivalent components in a software project without problems.

Config area A DD object that is a child object of the DD root object. The Config object contains configuration data for the tools working with the TargetLink Data Dictionary and configuration data for the TargetLink Data Dictionary itself. There is only one Config object in each DD workspace. The configuration data for the TargetLink Data Dictionary is a list of included DD files, user-defined views, data for variant configurations, etc. The data in the Config area is typically maintained by a Data Dictionary administrator.

ConfigurationDesk A dSPACE software tool for implementing and building real-time applications (RTA).

Constant value expression An expression for which the Code Generator can determine the variable values during code generation.

Constrained range limits User-defined minimum (Min) or maximum (Max) values that the user ensures will never be exceeded. The Code Generator relies on these ranges to make the generated production code more efficient. If no

Min/Max values are entered, the implemented range limits are used during production code generation.

Constrained type A DD Typedef object whose Constraints subtree is specified.

Container A bundle of files. The files are described in a catalog file that is part of the container. The files of a container can be spread over your file system.

Container Manager A tool for handling containers.

Container set file (CTS) A file that lists a set of containers. If you export containers, one container set file is created for every TargetLink Data Dictionary.

Conversion method A method that describes the conversion of a variable's integer values in the ECU memory into their physical representations displayed in the Measurement and Calibration (MC) system.

Custom code Custom code consists of C code snippets that can be included in production code by using custom code files that are associated with custom code blocks. TargetLink treats this code as a black box. Accordingly, if this code contains custom code variables you must specify them via custom code symbols.. See also external code.

Custom code symbol A variable that is used in a custom code file. It must be specified on the Interface page of custom code blocks.

Customer-specific C function An external function that is called from a Stateflow diagram and whose interface is made known to TargetLink via a scripting mechanism.

D

Data element Defined in a sender-receiver interface. Data elements are information units that are exchanged between sender ports, receiver ports and sender-receiver ports. They represent the data flow.

Data page A structure containing all of the calibratable variables that are generated during code generation.

Data prototype The generic term for one of the following:

- Data element
- Operation argument
- Calprm
- Interrunnable variable (IRV)
- Shared or PerInstance Calprm
- Per instance memory

Data receive error event An RTE event that specifies to start or continue the execution of a runnable related to receiver errors.

Data received event An RTE event that specifies whether to start or continue the execution of a runnable after a data element is received by a receiver port or sender-receiver port.

Data semantics The communication of data elements with last-is-best semantics. Newly received data elements overwrite older ones regardless of whether they have been processed or not.

Data send completed event An RTE event that specifies whether to start or continue the execution of a runnable related to a sender acknowledgment.

Data transformation A transformation of the data of inter-ECU communication, such as end-to-end protection or serialization, that is managed by the RTE via transformers.

Data type map Defines a mapping between implementation data types (represented in TargetLink by DD **Typedef** objects) and application data types.

Data type mapping set Summarizes all the data type maps and mode request type maps of a software component (SWC).

Data variant One of two or more differing data values that are generated into the same C code and can be switched during ECU run time using a calibratable variant ID variable. For example, the **Value** property of a gain parameter can have the variants 2, 3, and 4.

DataItemMapping (DIM) A DataItemMapping object is a DD object that references a ReplaceableDataItem (RDI) and a DD variable. It is used to define the DD variable object to map an RDI object to, and therefore also the implementation variable in the generated code.

DD child object The DD object below another DD object in the DD object tree.

DD data model The DD data model describes the object kinds, their properties and constraints as well as the dependencies between them.

DD file A DD file (*.dd) can be a DD project file or a partial DD file.

DD object Data item in the **Data Dictionary** that can contain DD child objects and DD properties.

DD object tree The tree that arranges all DD objects according to the DD data model.

DD project file A file containing the DD objects of a DD workspace.

DD root object The topmost DD object of the DD workspace.

DD subtree A part of the DD object tree containing a DD object and all its descendants.

DD workspace An independent organizational unit (central data container) and the largest entity that can be saved to file or loaded from a DD project file. Any number of DD workspaces is supported, but only the first (DD0) can be used for code generation.

Default enumeration constant Represents the default constant, i.e., the name of an enumerated value that is used for initialization if an initial value is required, but not explicitly specified.

Direct reuse The Code Generator adds the instance-specific variables to the reuse structure as leaf struct components.

E

ECU Abbreviation of *electronic control unit*.

ECU software The ECU software consists of all the software that runs on an ECU. It can be divided into the basic software, run-time environment (RTE), and the application layer.

ECU State Manager A piece of software that manages modes. An ECU state manager is part of the basic software.

Enhanceable Simulink block A Simulink® block that corresponds to a TargetLink simulation block, for example, the **Gain** block.

Enumerated value An enumerated value consists of an enumeration constant and a corresponding underlying integer value (enumeration value).

Enumeration constant An enumeration constant defines the name for an enumerated value.

Enumeration data type A data type with a specific name, a set of named enumerated values and a default enumeration constant.

Enumeration value An enumeration value defines the integer value for an enumerated value.

Event message Event messages are information units that are defined in a sender-receiver interface and exchanged between sender ports or receiver ports. They represent the control flow. On the receiver side, each event message is related to a buffer that queues the received messages.

Event semantics Communication of data elements with first-in-first-out semantics. Data elements are received in the same order they were sent. In simulations, TargetLink behaves as if data semantics was specified, even if you specified event semantics. However, TargetLink generates calls to the correct RTE API functions for data and event semantics.

ExchangeableWidth A DD object that defines code variants or improves code readability by using macros for signal widths.

Exclusive area Allows for specifying critical sections in the code that cannot preempt/interrupt each other. An exclusive area can be used to specify the mutual exclusion of runnables.

Executable application The generic term for offline simulation applications and real-time applications.

Explicit communication A communication mode in Classic AUTOSAR. The data is exchanged whenever data is required or provided.

Explicit object An explicit object is an object in production code that the Code Generator created from a direct specification made at a DD object or at a model element. For comparison, see implicit object.

Extern C Stateflow symbol A C symbol (function or variable) that is used in a Stateflow chart but that is defined in an external code module.

External code Existing C code files/modules from external sources (e.g., legacy code) that can be included by preprocessor directives and called by the C code generated by TargetLink. Unlike Custom code, external code is used as it is.

External container A container that is owned by the tool with that you are exchanging a software component but that is not the tool that triggers the container exchange. This container is used when you import files of a software component which were created or changed by the other tool.

F

Filter An algorithm that is applied to received data elements.

Fixed-Point Library A library that contains functions and macros for use in the generated production code.

Function AF The short form for an access function (AF) that is implemented as a C function.

Function algorithm object Generic term for either a MATLAB local function, the interface of a MATLAB local function or a local MATLAB variable.

Function class A class that represents group properties of functions that determine the function definition, function prototypes and function calls of a function in the generated production code. There are two types of function classes: predefined function class objects defined in the `/Pool/FunctionClasses` group in the DD and implicit function classes (default function classes) that can be influenced by templates in the DD.

Function code Code that is generated for a modular unit that represents functionality and can have abstract interfaces to be reused without changes in different contexts, e.g. in different integration models.

Function inlining The process of replacing a function call with the code of the function body during code generation by TargetLink via inline expansion. This reduces the function call overhead and enables further optimizations at the potential cost of larger code size.

Function interface An interface that describes how to pass the inputs and outputs of a function to the generated production code. It is described by the function signature.

Function subsystem A subsystem that is atomic and contains a Function block. When generating code, TargetLink generates it as a C function.

Functional Mock-up Unit (FMU) An archive file that describes and implements the functionality of a model based on the Functional Mock-up Interface (FMI) standard.

G

Global data store The specification of a DD DataStoreMemoryBlock object that references a variable and is associated with either a Simulink.Signal object or Data Store Memory block. The referenced variable must have a module specification and a fixed name and must be global and non-static. Because of its central specification in the Data Dictionary, you can use it across the boundaries of CGUs.

I

Implementation Describes how a specific internal behavior is implemented for a given platform (microprocessor type and compiler). An implementation mainly consists of a list of source files, object files, compiler attributes, and dependencies between the make and build processes.

Implementation data type (IDT) According to AUTOSAR, implementation data types are used to define types on the implementation level of abstraction. From the implementation point of view, this regards the storage and manipulation of digitally represented data. Accordingly, implementation data types have data semantics and do consider implementation details, such as the data type.

Implementation data types can be constrained to change the resolution of the digital representation or define a range that is to be considered. Typically, they correspond to typedef statements in C code and still abstract from platform specific details such as endianness.

See also application data type (ADT).

Implementation variable A variable in the generated production code to which a ReplaceableDataItem (RDI) object is mapped.

ImplementationPolicy A property of data element and Calprm elements that specifies the implementation strategy for the resulting variables with respect to consistency.

Implemented range The range of a variable defined by its scaling parameters. To avoid overflows, the implemented range must include the maximum and minimum values the variable can take in the simulation application and in the ECU.

Implicit communication A communication mode in Classic AUTOSAR. The data is exchanged at the start and end of the runnable that requires or provides the data.

Implicit object Any object created for the generated code by the TargetLink Code Generator (such as a variable, type, function, or file) that may not have been specified explicitly via a TargetLink block, a Stateflow object, or the TargetLink Data Dictionary. Implicit objects can be influenced via DD templates. For comparison, see explicit object.

Implicit property If the property of a DD object or of a model based object is not directly specified at the object, this property is created by the Code Generator and is based on internal templates or DD **Template** objects. These properties are called implicit properties. Also see implicit object and explicit object.

Included DD file A partial DD file that is inserted in the proper point of inclusion in the DD object tree.

Incremental code generation unit (CGU) Generic term for code generation units (CGUs) for which you can incrementally generate code. These are:

- Referenced models
- Subsystems configured for incremental code generation

Incremental CGUs can be nested in other model-based CGUs.

Indirect reuse The Code Generator adds pointers to the reuse structure which reference the indirectly reused instance-specific variables.

Indirect reuse has the following advantages to direct reuse:

- The combination of shared and instance-specific variable.
- The reuse of input/output variables of neighboring blocks.

Inline expansion The process of replacing a function call with the code of the function body. See also function inlining and compiler inlining.

Instance-specific variable A variable that is accessed by one reusable system instance. Typically, instance-specific variables are used for states and parameters whose value are different across instances.

Instruction set simulator (ISS) A simulation model of a microprocessor thatcan execute binary code compiled for the corresponding microprocessor. This allows the ISS to behave in the same way as the simulated microprocessor.

Integration model A model or TargetLink subsystem that contains modular units which it integrates to make a larger entity that provides its functionality.

Interface Describes the data elements, NvData, event messages, operations, or calibration parameters that are provided or required by a software component (SWC) via a port (AUTOSAR).

Internal behavior An element that represents the internal structure of an atomic software component (atomic SWC). It is characterized by the following entities and their interdependencies:

- Exclusive area
- Interrunnable variable (IRV)
- Per instance memory
- Per instance parameter
- Runnable
- RTE event
- Shared parameter

Interrunnable variable (IRV) Variable object for specifying communication between the runnables in one atomic software component (atomic SWC).

Interrupt service routine (ISR) function A function that implements an ISR and calls the step functions of the subsystems that are assigned by the user or by the TargetLink Code Generator during multirate code generation.

Intertask communication The flow of data between tasks and ISRs, tasks and tasks, and between ISRs and ISRs for multirate code generation.

Is service A property of an interface that indicates whether the interface is provided by a basic software service.

ISV Abbreviation for instance-specific variable.

L

Leaf bus element A leaf bus element is a subordinate bus element that is not a bus itself.

Leaf bus signal See also leaf bus element.

Leaf struct component A leaf struct component is a subordinate struct component that is not a struct itself.

Legacy function A function that contains a user-provided C function.

Library subsystem A subsystem that resides in a Simulink® library.

Local container A container that is owned by the tool that triggers the container exchange.

The tool that triggers the exchange transfers the files of a software component to this container when you export a software component. The external container is not involved.

Local MATLAB variable A variable that is generated when used on the left side of an assignment or in the interface of a MATLAB local function. TargetLink does not support different data types and sizes on local MATLAB variables.

Look-up function A function for a look-up table that returns a value from the look-up table (1-D or 2-D).

M

Macro A literal representing a C preprocessor definition. Macros are used to provide a fixed sequence of computing instructions as a single program statement. Before code compilation, the preprocessor replaces every occurrence of the macro by its definition, i.e., by the code that it stands for.

Macro AF The short form for an access function (AF) that is implemented as a function-like preprocessor macro.

MATLAB code elements MATLAB code elements include MATLAB local functions and local MATLAB variables. MATLAB code elements are not available in the Simulink Model Explorer or the Property Manager.

MATLAB local function A function that is scoped to a MATLAB main function and located at the same hierarchy level. MATLAB local functions are treated like MATLAB main functions and have the same properties as the MATLAB main function by default.

MATLAB main function The first function in a MATLAB function file.

Matrix AF An access function resulting from a DD AccessFunction object whose VariableKindSpec property is set to `APPLY_TO_MATRIX`.

Matrix signal Collective term for 2-D signals implemented as matrix variable in production code.

Matrix variable Collective term for 2-D arrays in production code that implement 2-D signals.

Measurement Viewing and analyzing the time traces of calibration parameters and measurement variables, for example, to observe the effects of ECU parameter changes.

Measurement and calibration system A tool that provides access to an ECU for measurement and calibration. It requires information on the calibration parameters and measurement variables with the ECU code.

Measurement variable Any variable type that can be measured but not calibrated. The term *measurement variable* is independent of a variable type's dimension.

Memory mapping The process of mapping variables and functions to different memory sections.

Memory section A memory location to which the linker can allocate variables and functions.

Message Browser A TargetLink component for handling fatal (F), error (E), warning (W), note (N), and advice (A) messages.

MetaData files Files that store metadata about code generation. The metadata of each code generation unit (CGU) is collected in a DD Subsystem object that is written to the file system as a partial DD file called `<CGU>_SubsystemObject.dd`.

Method Behavior subsystem An atomic subsystem used to generate code for a method implementation. From the TargetLink perspective, this is an Adaptive AUTOSAR Function that can take arguments.

It contains a Function block whose AUTOSAR mode is set to `Adaptive` and whose Role is set to `Method Behavior`.

Method Call subsystem An atomic subsystem that is used to generate a method call in the code of an Adaptive AUTOSAR Function. The subsystem contains a Function block whose AUTOSAR mode is set to `Adaptive` and whose Role is set to `Method Call`. The subsystem interface is used to generate the function interface while additional model elements that are contained in the subsystem are only for simulation purposes.

Microcontroller family (MCF) A group of microcontroller units with the same processor, but different peripherals.

Microcontroller unit (MCU) A combination of a specific processor with additional peripherals, e.g. RAM or AD converters. MCUs with the same processor, but different peripherals form a microcontroller family.

MIL simulation A simulation method in which the function model is computed (usually with double floating-point precision) on the host computer as an executable specification. The simulation results serve as a reference for SIL simulations and PIL simulations.

MISRA Organization that assists the automotive industry to produce safe and reliable software, e.g., by defining guidelines for the use of C code in automotive electronic control units or modeling guidelines.

Mode An operating state of an ECU, a single functional unit, etc..

Mode declaration group Contains the possible operating states, for example, of an ECU or a single functional unit.

Mode manager A piece of software that manages modes. A mode manager can be implemented as a software component (SWC) of the application layer.

Mode request type map An entity that defines a mapping between a mode declaration group and a type. This specifies that mode values are instantiated in the software component (SWC)'s code with the specified type.

Mode switch event An RTE event that specifies to start or continue the execution of a runnable as a result of a mode change.

Model Compare A dSPACE software tool that identifies and visualizes the differences in the contents of Simulink/TargetLink models (including Stateflow). It can also merge the models.

Model component A model-based code generation unit (CGU).

Model element A model in MATLAB/Simulink consists of model elements that are TargetLink blocks, Simulink blocks, and Stateflow objects, and signal lines connecting them.

Model port A port used to connect a behavior model in ConfigurationDesk. In TargetLink, multiple model ports of the same kind (data in or data out) can be grouped in a model port block.

Model port block A block in ConfigurationDesk that has one or more model ports. It is used to connect the behavior model in ConfigurationDesk.

Model port variable A DD Variable object that represents a model port of a behavior model in ConfigurationDesk.

Model-dependent code elements Code elements that (partially) result from specifications made in the model.

Model-independent code elements Code elements that can be generated from specifications made in the Data Dictionary alone.

Modular unit A submodel containing functionality that is reusable and can be integrated in different integration models. The production code for the modular unit can be generated separately.

Module A DD object that specifies code modules, header files, and other arbitrary files.

Module specification The reference of a DD Module object at a **Function Block** (TargetLink Model Element Reference) block or DD object. The resulting code elements are generated into the module. See also production code and stub code.

ModuleOwnership A DD object that specifies an owner for a module (module owner) or module group, i.e. the owning code generation unit (CGU) that generates the production code for it or declares the module as external code that is not generated by TargetLink.

N

Nested bus A nested bus is a bus that is a subordinate bus element of another bus.

Nested struct A nested struct is a struct that is a subordinate struct component of another struct.

Non-scalar signal Collective term for vector and matrix signals.

Non-standard scaling A scaling whose LSB is different from 2^0 `or whose Offset is not 0.`

Nv receiver port A require port in NvData communication as described by Classic AUTOSAR. In the Data Dictionary, nv receiver ports are represented as DD NvReceiverPort objects.

Nv sender port A provide port in NvData communication as described by Classic AUTOSAR. In the Data Dictionary, nv sender ports are represented as DD NvSenderPort objects.

Nv sender-receiver port A provide-require port in NvData communication as described by Classic AUTOSAR. In the Data Dictionary, nv sender-receiver ports are represented as DD NvSenderReceiverPort objects.

NvData Data that is exchanged between an atomic software component (atomic SWC) and the ECU's NVRAM.

NvData interface An interface used in NvData communication.

NVRAM Abbreviation of *non volatile random access memory*.

NVRAM manager A piece of software that manages an ECU's NVRAM. An NVRAM manager is part of the basic software.

O

Offline simulation application (OSA) An application that can be used for offline simulation in VEOS.

Online parameter modification The modification of parameters in the production code before or during a SIL simulation or PIL simulation.

Operation Defined in a client-server interface. A software component (SWC) can request an operation via a client port. A software component can provide an operation via a server port. Operations are implemented by server runnables.

Operation argument Specifies a C-function parameter that is passed and/or returned when an operation is called.

Operation call subsystem A collective term for synchronous operation call subsystem and asynchronous operation call subsystem.

Operation call with runnable implementation subsystem An atomic subsystem that contains a Function block whose AUTOSAR mode property is set to `Classic` and whose Role is set to `Operation call with runnable implementation`.

Operation invoked event An RTE event that specifies to start or continue the execution of a runnable as a result of a client call. A runnable that is related to an operation invoked event represents a server.

Operation result provider subsystem A subsystem used when modeling *asynchronous* client-server communication. It is used to generate the call of the `Rte_Result` API function and for simulation purposes.

See also asynchronous operation call subsystem.

Operation subsystem A collective term for operation call subsystem and operation result provider subsystem.

OSEK Implementation Language (OIL) A modeling language for describing the configuration of an OSEK application and operating system.

P

Package A structuring element for grouping elements of software components in any hierarchy. Using package information, software components can be spread across or combined from several software component description (SWC-D) files during AUTOSAR import/export scenarios.

Parent model A model containing references to one or more other models by means of the Simulink Model block.

Partial DD file A DD file that contains only a DD subtree. If it is included in a DD project file, it is called Included DD file. The partial DD file can be located on a central network server where all team members can share the same configuration data.

Per instance memory The definition of a data prototype that is instantiated for each atomic software component instance by the RTE. A data type instance can be accessed only by the corresponding instance of the atomic SWC.

Per instance parameter A parameter for measurement and calibration unique to the instance of a software component (SWC) that is instantiated multiple times.

Physical evaluation board (physical EVB) A board that is equipped with the same target processor as the ECU and that can be used for validation of the generated production code in PIL simulation mode.

PIL simulation A simulation method in which the TargetLink control algorithm (production code) is computed on a microcontroller target (physical or virtual).

Plain data type A data type that is not struct, union, or pointer.

Platform A specific target/compiler combination. For the configuration of platforms, refer to the Code generation target settings in the TargetLink Main Dialog Block block.

Pool area A DD object which is parented by the DD root object. It contains all data objects which can be referenced in TargetLink models and which are used for code generation. Pool data objects allow common data specifications to be reused across different blocks or models to easily keep consistency of common properties.

Port (AUTOSAR) A part of a software component (SWC) that is the interaction point between the component and other software components.

Port-defined argument values Argument values the RTE can implicitly pass to a server.

Preferences Editor A TargetLink tool that lets users view and modify all user-specific preference settings after installation has finished.

Production code The code generated from a code generation unit (CGU) that owns the module containing the code. See also stub code.

Project folder A folder in the file system that belongs to a TargetLink code generation project. It forms the root of different artifact locations that belong to this project.

Property Manager The TargetLink user interface for conveniently managing the properties of multiple model elements at the same time. It can consist of menus, context menus, and one or more panes for displaying property–related information.

Provide calprm port A provide port in parameter communication as described by Classic AUTOSAR. In the Data Dictionary, provide calprm ports are represented as DD ProvideCalPrmPort objects.

R

Read/write access function An access function (AF) that *encapsulates the instructions* for reading or writing a variable.

Real-time application An application that can be executed in real time on dSPACE real-time hardware such as SCALEXIO.

Receiver port A require port in sender-receiver communication as described by Classic AUTOSAR. In the Data Dictionary, receiver ports are represented as DD ReceiverPort objects.

ReplaceableDataItem (RDI) A ReplaceableDataItem (RDI) object is a DD object that describes an abstract interface's basic properties such as the data type, scaling and width. It can be referenced in TargetLink block dialogs and is generated as a global macro during code generation. The definition of the RDI macro can then be generated later, allowing flexible mapping to an implementation variable.

Require calprm port A require port in parameter communication as described by Classic AUTOSAR. In the Data Dictionary, require calprm ports are represented as DD RequireCalPrmPort objects.

RequirementInfo An object of a DD RequirementInfo object. It describes an item of requirement information and has the following properties: Description, Document, Location, UserTag, ReferencedInCode, SimulinkStateflowPath.

Restart function A production code function that initializes the global variables that have an entry in the RestartfunctionName field of their variable class.

Reusable function definition The function definition that is to be reused in the generated code. It is the code counterpart to the reusable system definition in the model.

Reusable function instance An instance of a reusable function definition. It is the code counterpart to the reusable system instance in the model.

Reusable model part Part of the model that can become a reusable system definition. Refer to Basics on Function Reuse (TargetLink Customization and Optimization Guide).

Reusable system definition A model part to which the function reuse is applied.

Reusable system instance An instance of a reusable system definition.

Root bus A root bus is a bus that is not a subordinate part of another bus.

Root function A function that represents the starting point of the TargetLink-generated code. It is called from the environment in which the TargetLink-generated code is embedded.

Root model The topmost parent model in the system hierarchy.

Root module The module that contains all the code elements that belong to the production code of a code generation unit (CGU) and do not have their own module specification.

Root step function A step function that is called only from outside the production code. It can also represent a non-TargetLink subsystem within a TargetLink subsystem.

Root struct A root struct is a struct that is not a subordinate part of another struct.

Root style sheet A root style sheet is used to organize several style sheets defining code formatting.

RTE event The abbreviation of run-time environment event.

Runnable A part of an atomic SWC. With regard to code execution, a runnable is the smallest unit that can be scheduled and executed. Each runnable is implemented by one C function.

Runnable execution constraint Constraints that specify runnables that are allowed or not allowed to be started or stopped before a runnable.

Runnable subsystem An atomic subsystem that contains a Function block whose AUTOSAR mode property is set to `Classic` and whose Role is set to `Runnable`.

Run-time environment (RTE) A generated software layer that connects the application layer to the basic software. It also interconnects the different SWCs of the application layer. There is one RTE per ECU.

Run-time environment event A part of an internal behavior. It defines the situations and conditions for starting or continuing the execution of a specific runnable.

S

Scaling A parameter that specifies the fixed-point range and resolution of a variable. It consists of the data type, least significant bit (LSB) and offset.

Sender port A provide port in sender-receiver communication as described by Classic AUTOSAR. In the Data Dictionary, sender ports are represented as DD SenderPort objects.

Sender-receiver interface An interface that describes the data elements and event messages that are provided or required by a software component (SWC) via a port (AUTOSAR).

Sender-receiver port A provide-require port in sender-receiver communication as described by Classic AUTOSAR. In the Data Dictionary, sender-receiver ports are represented as DD SenderReceiverPort objects.

Server port A provide port in client-server communication as described by Classic AUTOSAR. In the Data Dictionary, server ports are represented as DD ServerPort objects.

Server runnable A runnable that provides an operation via a server port. Server runnables are triggered by operation invoked events.

Shared parameter A parameter for measurement and calibration that is used by several instances of the same software component (SWC).

Shared variable A variable that is accessed by several reusable system instances. Typically, shared variables are used for parameters whose values are the same across instances. They increase code efficiency.

SIC runnable function A void (void) function that is called in a task. Generated into the Simulink implementation container (SIC) to call the root function that is generated by TargetLink from a TargetLink subsystem. In ConfigurationDesk, this function is called *runnable function*.

SIL simulation A simulation method in which the control algorithm's generated production code is computed on the host computer in place of the corresponding model.

Simple TargetLink model A simple TargetLink model contains at least one TargetLink Subsystem block and exactly one MIL Handler block.

Simplified initialization mode The initialization mode used when the Simulink diagnostics parameter Underspecified initialization detection is set to `Simplified`.

See also classic initialization mode.

Simulation application An application that represents a graphical model specification (implemented control algorithm) and simulates its behavior in an offline Simulink environment.

Simulation code Code that is required only for simulation purposes. Does not belong to the production code.

Simulation S-function An S-function that calls either the root step functions created for a TargetLink subsystem, or a user-specified step function (only possible in test mode via API).

Simulink data store Generic term for a memory region in MATLAB/Simulink that is defined by one of the following:

- A Simulink.Signal object
- A Simulink Data Store Memory block

Simulink function call The location in the model where a Simulink function is called. This can be:

- A Function Caller block
- The action language of a Stateflow Chart
- The MATLAB code of a MATLAB function

Simulink function definition The location in the model where a Simulink function is defined. This can be one of the following:

- Simulink Function subsystem
- Exported Stateflow graphical function
- Exported Stateflow truthtable function
- Exported Stateflow MATLAB function

Simulink function ports The ports that can be used in a Simulink Function subsystem. These can be the following:

- TargetLink ArgIn and ArgOut blocks

  These ports are specific for each Simulink function call.
- TargetLink InPort/OutPort and Bus Inport/Bus Outport blocks

  These ports are the same for all Simulink function calls.

Simulink Function subsystem A subsystem that contains a Trigger block whose Trigger Type is `function-call` and whose Treat as Simulink Function checkbox is selected.

Simulink implementation container (SIC) A file that contains all the files required to import production code generated by TargetLink into ConfigurationDesk as a behavior model with model ports.

Slice A section of a vector or matrix signal, whose elements have the same properties. If all the elements of the vector/matrix have the same properties, the whole vector/matrix forms a slice.

Software component (SWC) The generic term for atomic software component (atomic SWC), compositions, and special software components, such as calprm software components. A software component logically groups and encapsulates single functionalities. Software components communicate with each other via ports.

Software component description (SWC-D) An XML file that describes software components according to AUTOSAR.

Stateflow action language The formal language used to describe transition actions in Stateflow.

Struct A struct (short form for structure) consists of subordinate struct components. A struct component can be a struct itself.

Struct component A struct component is a part of a struct and can be a struct itself.

Structure A structure (long form for struct) consists of subordinate struct components. A struct component can be a struct itself.

Stub code Code that is required to build the simulation application but that belongs to another code generation unit (CGU) than the one used to generate production code.

Subsystem area A DD object which is parented by the DD root object. This object consists of an arbitrary number of Subsystem objects, each of which is the result of code generation for a specific code generation unit (CGU). The Subsystem objects contain detailed information on the generated code, including C modules, functions, etc. The data in this area is either automatically generated or imported from ASAM MCD-2 MC, and must not be modified manually.

Supported Simulink block A TargetLink-compliant block from the Simulink library that can be directly used in the model/subsystem for which the Code Generator generates production code.

SWC container A container for files of one SWC.

Synchronous operation call subsystem A subsystem used when modeling *synchronous* client-server communication. It is used to generate the call of the `Rte_Call` API function and for simulation purposes.

T

Table function A function that returns table output values calculated from the table inputs.

Target config file An XML file named `TargetConfig.xml`. It contains information on the basic data types of the target/compiler combination such as the byte order, alignment, etc.

Target Optimization Module (TOM) A TargetLink software module for optimizing production code generation for a specific microcontroller/compiler combination.

Target Simulation Module (TSM) A TargetLink software module that provides support for a number of evaluation board/compiler combinations. It is used to test the generated code on a target processor. The TSM is licensed separately.

TargetLink AUTOSAR Migration Tool A software tool that converts classic, non-AUTOSAR TargetLink models to AUTOSAR models at a click.

TargetLink AUTOSAR Module A TargetLink software module that provides extensive support for modeling, simulating, and generating code for AUTOSAR software components.

TargetLink Base Suite The base component of the TargetLink software including the ANSI C Code Generator and the Data Dictionary Manager.

TargetLink base type One of the types used by TargetLink instead of pure C types in the generated code and the delivered libraries. This makes the code platform independent.

TargetLink Blockset A set of blocks in TargetLink that allow production code to be generated from a model in MATLAB/Simulink.

TargetLink Data Dictionary The central data container thats holds all relevant information about an ECU application, for example, for code generation.

TargetLink simulation block A block that processes signals during simulation. In most cases, it is a block from standard Simulink libraries but carries additional information required for production code generation.

TargetLink subsystem A subsystem from the TargetLink block library that defines a section of the Simulink model for which code must be generated by TargetLink.

Task A code section whose execution is managed by the real-time operating system. Tasks can be triggered periodically or based on events. Each task can call one or more SIC runnable functions.

Task function A function that implements a task and calls the functions of the subsystems which are assigned to the task by the user or via the TargetLink Code Generator during multirate code generation.

Term function A function that contains the code to be executed when the simulation finishes or the ECU application terminates.

Terminate function A runnable that finalizes a SWC, for example, by calling code that has to run before the application shuts down.

Timing event An RTE event that specifies to start or continue the execution of a runnable at constant time intervals.

tllib A TargetLink block library that is the source for creating TargetLink models graphically. Refer to How to Open the TargetLink Block Library (TargetLink Orientation and Overview Guide).

Transformer The Classic AUTOSAR entity used to perform a data transformation.

TransformerError The parameter passed by the run-time environment (RTE) if an error occurred in a data transformation. The `Std_TransformerError` is a struct whose components are the transformer class and the error code. If the error is a hard error, a special runnable is triggered via the TransformerHardErrorEvent to react to the error.

In AUTOSAR releases prior to R19-11 this struct was named `Rte_TransformerError`.

TransformerHardErrorEvent The RTE event that triggers the runnable to be used for responding to a hard TransformerError in a data transformation for client-server communication.

Type prefix A string written in front of the variable type of a variable definition/declaration, such as `MyTypePrefix Int16 MyVar`.

U

Unicode The most common standard for extended character sets is the Unicode standard. There are different schemes to encode Unicode in byte format, e.g., UTF-8 or UTF-16. All of these encodings support all Unicode characters. Scheme conversion is possible without losses. The only difference between these encoding schemes is the memory that is required to represent Unicode characters.

User data type (UDT) A data type defined by the user. It is placed in the Data Dictionary and can have associated constraints.

Utility blocks One of the categories of TargetLink blocks. The blocks in the category keep TargetLink-specific data, provide user interfaces, and control the simulation mode and code generation.

V

Validation Summary Shows unresolved model element data validation errors from all model element variables of the Property View. It lets you search, filter, and group validation errors.

Value copy AF An access function (AF) resulting from DD AccessFunction objects whose AccessFunctionKind property is set to `READ_VALUE_COPY` or `WRITE_VALUE_COPY`.

Variable access function An access function (AF) that *encapsulates the access* to a variable for reading or writing.

Variable class A set of properties that define the role and appearance of a variable in the generated production code, e.g. CAL for global calibratable variables.

VariantConfig A DD object in the Config area that defines the code variants and data variants to be used for simulation and code generation.

VariantItem A DD object in the DD Config area used to variant individual properties of DD Variable and ExchangeableWidth objects. Each variant of a property is associated with one variant item.

V-ECU implementation container (VECU) A file that consists of all the files required to build an offline simulation application (OSA) to use for simulation with VEOS.

V-ECU Manager A component of TargetLink that allows you to configure and generate a V-ECU implementation.

Vendor mode The operation mode of RTE generators that allows the generation of RTE code which contains vendor-specific adaptations, e.g., to reduce resource consumption. To be linkable to an RTE, the object code of an SWC must have been compiled against an application header that matches the RTE code generated by the specific RTE generator. This is the case because the data structures and types can be implementation-specific.

See also compatibility mode.

VEOS A dSPACE software platform for the C-code-based simulation of virtual ECUs and environment models on a PC.

Virtual ECU (V-ECU) Software that emulates a real ECU in a simulation scenario. The virtual ECU comprises components from the application and the basic software, and provides functionalities comparable to those of a real ECU.

Virtual ECU testing Offline and real-time simulation using virtual ECUs.

Virtual evaluation board (virtual EVB) A combination of an instruction set simulator (ISS) and a simulated periphery. This combination can be used for validation of generated production code in PIL simulation mode.

W

Worst-case range limits A range specified by calculating the minimum and maximum values which a block's output or state variable can take on with respect to the range of the inputs or the user-specified constrained range limits.

Troubleshooting

Introduction	When working with the TargetLink Data Dictionary, you should be aware of some limitations and problems you might experience.

Data Dictionary Troubleshooting

Introduction	Certain technical questions might arise when working with Data Dictionaries.
Correct DD settings	*I am not sure if my DD settings are correct.* ▪ Perform a level 4 validation of the Data Dictionary, ▪ In the Data Dictionary Manager, select **Validate** from the context menu of the root object. ▪ In the MATLAB Command Window, use the command `dsdd_validate('//DD0','ValidationLevel', 4)`.
Wrong DD project file loaded	*The wrong DD project file is loaded when a Simulink model is opened.* The global preferences for the DD project file name are not set correctly. ▪ Open the Preferences Editor with the MATLAB API function `tl_pref` and set the preferred DD project file name needed for your model. You can also associate the model with a fixed DD project file name on the **Options** page of the **TargetLink Main Dialog Block**.
Another DD project file loaded	*Another DD project file is loaded when a second Simulink model is opened.* The second Simulink model is associated with a fixed DD project file name. ▪ Only one DD project file can be open for code generation in DD0. When two models are open, they must use the same DD project file.

MATLAB hangs on exit

MATLAB seems to hang when I want to exit.

If you shut down MATLAB and the current Data Dictionary has been modified, a message box queries whether you want to save it. This message box might disappear in the background if numerous windows are open.

- Check your taskbar for the message box. Move it to the foreground and decide whether you wish to save the current DD.

File I/O error when saving DD project file

When exiting MATLAB, I clicked "yes" when asked whether to save the current DD. However, I get another message box displaying file I/O error messages.

The current Data Dictionary could not be saved during shutdown, for example, because there was no write access to the associated DD files.

- Leave the error message box open. Check the files and paths involved. Try to fix the problems, for example, by modifying file attributes. Click **Retry** for another attempt. If the problem cannot be solved, click **Cancel**. However, this leaves the DD unsaved.

Modifications not reflected in Property Manager

I have modified some objects in the /Pool area, but the modifications are not reflected in TargetLink's Property Manager.

- On the **Home** ribbon of the Property Manager, click **Data Dictionary Resync** in the **Model Element Data** ribbon group.

Expected messages do not show up

I am working with the Data Dictionary and expect some messages. However, some messages never seem to show up.

Some messages are suppressed.

- Do not suppress messages if you are working with the Data Dictionary. This feature should be used very cautiously for a limited number of use cases. Many functions rely on Data Dictionary messaging to work reliably.

User-defined property appears to be double

I have written a user-defined property to an object. The property should be Boolean. However, in the Data Dictionary Manager it appears to be double.

For user-defined properties, i.e., properties not defined in the data model, the DD MATLAB API cannot know the data type, but assigns the type of the specified value. For numeric scalars, this is usually double. If you wish to have another type assigned, it must be explicitly specified.

More error messages than errors

An error has come up. However, the Message Browser displays more than one DD error message.

To ensure that you are shown all relevant data, all possible error messages are issued, some of which are redundant. As long as you are informed of the reason and solution, and help on solving the problem is provided, you can simply ignore redundant messages.

Related topics

References

TargetLink Main Dialog Block (📖 TargetLink Model Element Reference)