AutomationDesk

# Accessing Remote Diagnostics COM

For AutomationDesk 6.5

Release 2021-A – May 2021

dSPACE

## How to Contact dSPACE


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


## How to Contact dSPACE Support

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

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

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

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

## Software Updates and Patches

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

## Important Notice

# Contents

# About This Document

**Content**

This document gives you information on how to access a diagnostic tool via AutomationDesk.

**Required knowledge**

Working with AutomationDesk requires:

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

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

**Symbols**

dSPACE user documentation uses the following symbols:

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

| Symbol | Description |
|---|---|
| 🕮 | Precedes the document title in a link that refers to another document. |

**Naming conventions**

dSPACE user documentation uses the following naming conventions:

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

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

**Special folders**

Some software products use the following special folders:

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

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

or

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

**Documents folder** A standard folder for user-specific documents.

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

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

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

**Accessing dSPACE Help and PDF Files**

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

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

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

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

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

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

# Basics and Instructions

**Where to go from here**

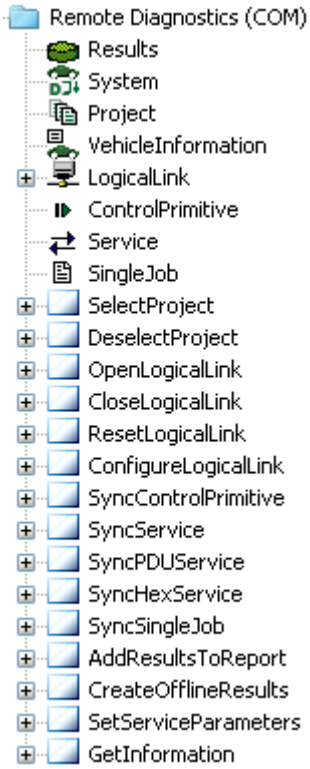
Information in this section

# Overview of the Remote Diagnostics (COM) Library

**Library overview**

**Note**

AutomationDesk's Remote Diagnostics (COM) library supports ControlDesk 5.1 or later as the diagnostic tools based on ASAM MCD-3 D 2.0.2.



## Data objects

**Results** The Results data object contains the diagnostic results that are delivered by the diagnostic tool. These results are only available during the execution of the project. If you have used the CreateOfflineResults automation block, the diagnostic results are stored to the AutomationDesk project as Python object. For further information, refer to Results on page 50.

**Hierarchical representation of the diagnostic data** All the other data objects in the library are used to configure the diagnostic tool and the required diagnostic tasks:

- System - contains the interface parameters of the diagnostic tool.

  Refer to System on page 59.
- Project - provides a list of all the available diagnostic projects of the connected diagnostic system.

  Refer to Project on page 48.
- Vehicle Information - describes which ECUs are installed in the vehicle. It can contain one or more LogicalLink data objects.

  Refer to VehicleInformation on page 60.
- LogicalLink - sets up the connection to the ECU. The LogicalLink data object provides all communication primitives, services, and single ECU jobs of the ECU.

  Refer to LogicalLink on page 46.
- ControlPrimitive - holds the available communication primitives of the selected logical link.

  Refer to ControlPrimitive on page 43.
- Service - holds the parameters of the selected diagnostic service. The available services are provided by the selected logical link.

  Refer to Service on page 51.
- SingleJob - holds the parameters of the selected single ECU job. The available jobs are provided by the selected logical link.

  Refer to SingleJob on page 53.

## Automation blocks

The Remote Diagnostics (COM) library provides automation blocks for:

- Handling the diagnostic project.

  Refer to SelectProject on page 50 and DeselectProject on page 45.
- Handling logical links.

  Refer to OpenLogicalLink on page 47, CloseLogicalLink on page 40 and ResetLogicalLink on page 49.
- Reading diagnostic data from the ECU synchronously.

  Refer to SyncControlPrimitive on page 54, SyncService on page 58, SyncHexService on page 55, SyncPDUService on page 56 and SyncSingleJob on page 57
- Adding diagnostic results to a report.

  Refer to AddResultsToReport on page 39.
- Creating a storable diagnostic result in online mode.

  Refer to CreateOfflineResults on page 44.

- Modifying values of a service parameter during test execution.
  Refer to SetServiceParameters on page 52.
- Configuring the referenced logical link to the ECU during test execution.
  Refer to ConfigureLogicalLink on page 41.
- Getting information on the referenced diagnostic object.
  Refer to GetInformation on page 45.

**Related topics**

Examples



References



# Example of a Diagnostic Project Sequence

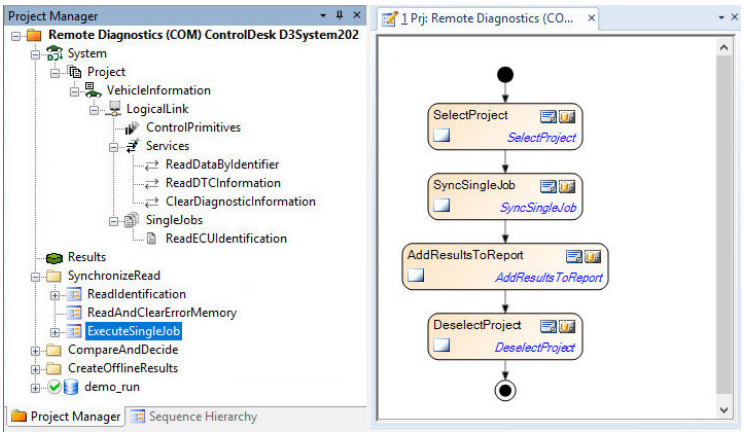
**Introduction**

The following automation sequence shows you a program to automate an ECU diagnostic task via diagnostic tool. Certain identifiers (parameters) are read from the ECU for further diagnostic use and their values are added to a report.

- SelectProject
  The project is selected and the connection to the ECU via diagnostic tool is set up.
- SyncSingleJob
  The SyncSingleJob block executes the specified diagnostic single ECU job from the ECU via diagnostic tool synchronously. The results are stored in the Results data object.
- AddResultsToReport (AddIdentificationToReport)
  The AddResultsToReport block adds the results that are stored in the Results data object to the report.

- DeselectProject

  The DeselectProject block deactivates the specified project and closes the connection between AutomationDesk and diagnostic tool.




**Note**

In this example, the data objects used by the automation blocks in the sequence are additionally created in the project structure. This allows you to change the references of sequence-specific data objects without editing the sequence itself. In some cases it is useful to do so, but it is not required.

**Demo projects**

Further AutomationDesk demo projects can be found at `<DocumentsFolder>\Remote Diagnostics (COM)`.

**Related topics**

Basics

Overview of the Remote Diagnostics (COM) Library.....8

HowTos

How to Add the Results to a Report.....34
How to Build a Basic Sequence For Diagnostic Tasks.....22
How to Read Data From the Diagnostic Tool Synchronously.....25

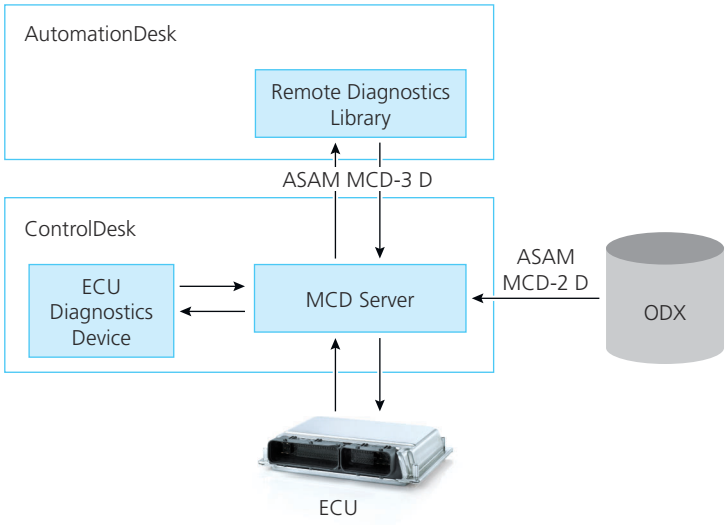
# Basics of Automating ControlDesk's Diagnostics Features via AutomationDesk

**Introduction**

You can automate diagnostic tasks that access an ECU by remotely controlling the Measurement Calibration and Diagnostics (MCD) server that is provided by ControlDesk.

**Overview on the involved components**

The following illustration shows how the ECU is accessed via ControlDesk's MDC server.




The AutomationDesk **Remote Diagnostics (COM)** library provides data objects and automation blocks to access the MCD server via the ASAM MCD-D 3 interface of the server. The diagnostic services to be implemented for the ECU and the way these services are provided is configured in the Open Diagnostic Data Exchange (ODX) database according to the ASAM MCD-2 D standard. In ControlDesk, MCD servers and ODX databases are handled by using **ECU Diagnostics** devices.

For more information on configuring ODX databases and creating diagnostic devices in ControlDesk experiments, refer to ECU Diagnostics Device Configuration (ControlDesk ECU Diagnostics 🕮).

For more information on using the ControlDesk MCD3 Interface for diagnostic purposes, refer to Basics on Automating ECU Diagnostics via ControlDesk's MCD3 Interface (ControlDesk MCD-3 Automation 🕮).

For more information on how diagnostic tasks can be automated via ControlDesk's ASAM MCD-3 D interface, refer to Automating ControlDesk's Diagnostics Features (ControlDesk MCD-3 Automation 🕮). In AutomationDesk, the **Remote Diagnostics (COM)** library facilitates the implementation of the automated diagnostic tasks.

**Related topics**

Basics

Overview of the Remote Diagnostics (COM) Library.....8

# How to Set Up Diagnostic Projects

**Objective**

Before you create an automation sequence for handling a diagnostic task via diagnostic tool, you should set up the hierarchical data structure according to the ASAM MCD-3 D standard in AutomationDesk, containing all communication primitives, services, and single ECU jobs you want to automate.

The data objects must be created in the Project Manager as project-specific parameters. The parameters of the automation blocks can be parameterized by referencing these project-specific data objects.

**Basics**

For basic information on the automation process of diagnostic tasks with ControlDesk, refer to Basics of Automating ControlDesk's Diagnostics Features via AutomationDesk on page 11.

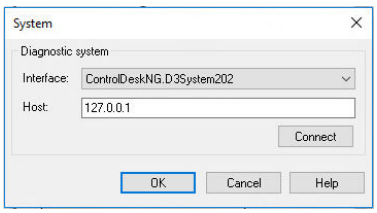
**Preconditions**

The diagnostic tool is installed and prepared.

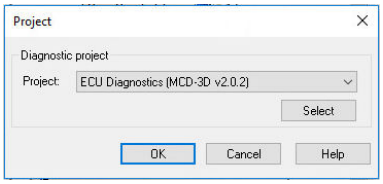
**Method**

**To set up a diagnostic project**

1. Create a new AutomationDesk project.
2. Open the Library Browser and drag a **System** data object from the Remote Diagnostics (COM) library to the AutomationDesk project element in the Project Manager.
3. Double-click the **System** data object in the Project Manager to open the **System** configuration dialog. Choose an interface from the list and specify the PC on which the diagnostic tool is installed by its name or IP address. *127.0.0.1* is used if it is installed on the local PC.



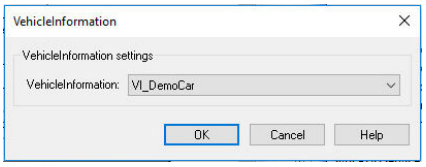

4. Click **Connect** to connect to the diagnostic system. If an error appears, check the preconditions. Click **OK** to close the dialog.
5. Choose **Add Project** from the System data object's context menu to add a diagnostic project to the Project Manager.
6. Double-click the **Project** data object in the Project Manager to open the **Project** configuration dialog. The project list displays all experiments of the ControlDesk projects that are contained in ControlDesk's project root folder. You can specify the ODX database to be used for communicating with the ECU by selecting the experiment that contains the related ECU diagnostic device. The ControlDesk experiment name matches the diagnostic project name.




Choose the diagnostic project name and click **Select** to load the diagnostic project's ODX database. Click **OK** to close the dialog.

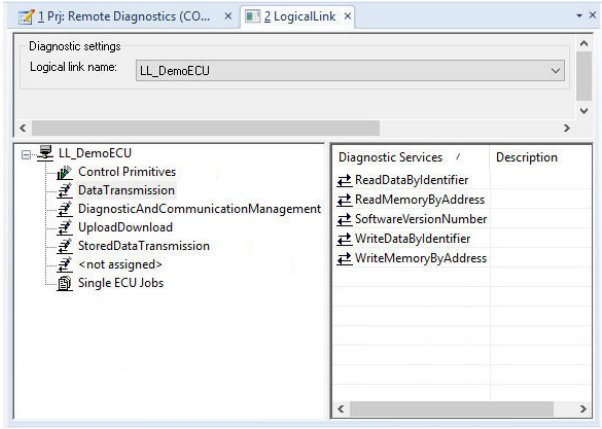
**7** Choose **Add VehicleInformation** from the Project data object's context menu to add a VehicleInformation data object to the Project Manager.

**8** Double-click the **VehicleInformation** data object to open the **VehicleInformation** configuration dialog. Choose an entry from the list. Its entries are provided by the diagnostic project's database. Click **OK** to close the dialog.

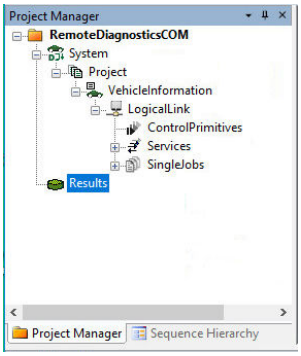



**9** Choose **Add LogicalLink** from the VehicleInformation data object's context menu to add a LogicalLink data object to the Project Manager. A **ControlPrimitives**, a **Services**, and a **SingleJobs** data container are also created. They can be used to group the communication primitives, services, and single ECU jobs of the diagnostic project.

**10** Double-click the **LogicalLink** data object to open the **LogicalLink** configuration dialog. Choose an entry from the list that is provided by the project's database. Click once in the browser window or the table to display the available communication primitives, services, and single ECU jobs. The services in the browser are grouped by functional classes. You cannot edit the entries in the table, but you can drag them to the Project Manager to create and parameterize ControlPrimitive, Service, and SingleJob data objects. Close the dialog by using the Close button in the dialog header.




**11** Drag a **Results** data object from the Library Browser to the AutomationDesk project element in the Project Manager. This project-specific Results data object is referenced by automation blocks that contain a Results data object.




**Result**

You have created project-specific data objects in the Project Manager.

**Note**

- To avoid malfunction, do not load a diagnostic experiment containing an ECU Diagnostics device during an automation session involving diagnostic tasks.
- If you replace the diagnostic project's database afterwards, you have to reconfigure the items in the configuration dialogs. For example, you have to reselect the Logical link name before you can choose a service or a job.
- Generally, this hierarchical structure of diagnostic data objects is only suitable for one ECU. When you test ECU variants, for example, it is not sufficient to change the logical link only. It is recommended to build separate data structures for variant testing.

**Next step**

In the next step you can specify the communication primitives, diagnostic services, and single ECU jobs which you want to manage via AutomationDesk, refer to:

- How to Configure ControlPrimitive Data Objects on page 16
- How to Configure Service Data Objects on page 18
- How to Configure SingleJob Data Objects on page 20

**Related topics**

Basics

Basics of Automating ControlDesk's Diagnostics Features via AutomationDesk..... 11

Examples



References



# How to Configure ControlPrimitive Data Objects

**Objective**

A ControlPrimitive data object contains a diagnostic communication primitive (ControlPrimitive) for communication with the ECU. After a connection to the diagnostic tool is set up, all the available parameters of the ControlPrimitive are listed in the **ControlPrimitive Configuration** dialog, where you can customize them.

The ControlPrimitive data objects must be created in the Project Manager as project-specific parameters. The ControlPrimitive data object of automation blocks must be parameterized by referencing these project-specific data objects.

**Preconditions**

- You have created the required data objects in the Project Manager, refer to How to Set Up Diagnostic Projects on page 13.
- AutomationDesk is connected to a diagnostic tool.

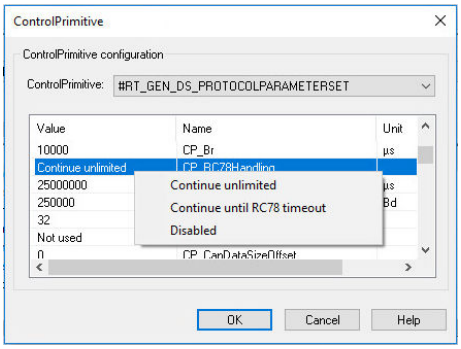
**Method**

**To configure the ControlPrimitive data object**

1 Add a **ControlPrimitive** data object to the **ControlPrimitives** data container in the Project Manager. You can do this in two different ways:
   - Open the **LogicalLink** configuration dialog and drag the required communication primitive to the **ControlPrimitives** data container. The created ControlPrimitive data object is parameterized with the values predefined in the diagnostic project and given the name of the ControlPrimitive.
   - Choose **Add ControlPrimitive** from the ControlPrimitives data container's context menu to add a ControlPrimitive data object. Double-click the ControlPrimitive data object to open the **ControlPrimitive** configuration

dialog. Choose an entry from the list to parameterize the data object with the required communication primitive.

2 Modify the values of the specified communication primitive. Open the **ControlPrimitive** configuration dialog by double-clicking the ControlPrimitive data object. If the communication primitive provides parameters, they are displayed with their values, names, and units. If there are predefined value ranges specified in the project's database, you can modify a value using the context menu of the parameter entry (right-click the value), otherwise you can enter any value.




**Tip**

Values which must be set in hexadecimal notation are displayed as `0x00`. Several successive hexadecimal entries have to be separated by commas. For example, `0xa1,f2,44`.

3 Click **OK** to confirm your settings and close the dialog.

**Result**

You have created and specified a ControlPrimitive data object that you can use as a data object of an automation block.

**Next step**

You can also specify diagnostic services to be automated, refer to How to Configure Service Data Objects on page 18.

**Related topics**

Basics



HowTos



Examples



References



# How to Configure Service Data Objects

**Objective**

A Service data object contains a diagnostic service for communication with the ECU. After a connection to the diagnostic tool is set up, all the available parameters of the Service are listed in the **Service** configuration dialog, where you can customize them.

The Service data objects must be created in the Project Manager as project-specific parameters. The Service data object of automation blocks must be parameterized by referencing these project-specific data objects.

**Preconditions**

- You have created the required data objects in the Project Manager, refer to How to Set Up Diagnostic Projects on page 13.
- AutomationDesk is connected to the diagnostic tool.

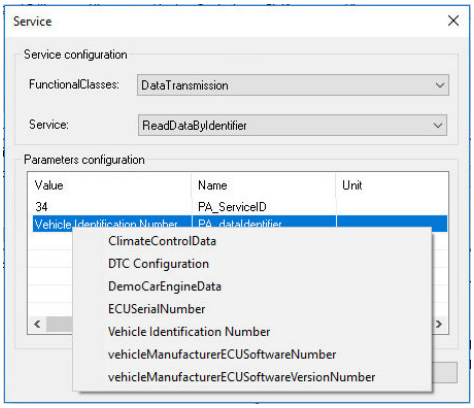
**Method**

**To configure the Service data object**

1 Add a **Service** data object to the **Services** data container in the Project Manager. You can do this in two different ways:
   - Open the **LogicalLink** configuration dialog and drag the required diagnostic service to the **Services** data container. The created **Service** data object is parameterized with the values predefined in the diagnostic project and given the name of the service.
   - Choose **Add Service** from the **Services** data container's context menu to add a **Service** data object. Double-click the **Service** data object to open the **Service** configuration dialog. Choose a functional class, which is used to

group the available services, and then select an entry from the service list to parameterize the data object with the required service.

2 Modify the values of the specified **Service** data object. Open the **Service** configuration dialog by double-clicking the Service data object. If the diagnostic service provides parameters, they are displayed with their values, names, and units. If there are predefined value ranges specified in the project's database, you can modify a value using the context menu of the parameter entry (right-click the value), otherwise you can enter any value.




**Tip**

Values which must be set in hexadecimal notation are displayed as `0x00`. Several successive hexadecimal entries have to be separated by commas. For example, `0xa1,f2,44`.

3 Click **OK** to confirm your settings and close the dialog.

**Result**

You have specified a Service data object that you can use as a data object of an automation block.

**Next step**

After you have specified all ControlPrimitive and Service data objects, you can also specify SingleJob data objects to be automated, refer to How to Configure SingleJob Data Objects on page 20.

**Related topics**

Basics



HowTos



Examples



References



# How to Configure SingleJob Data Objects

**Objective**

A SingleJob data object contains a diagnostic single ECU job for communication with the ECU. After a connection to the diagnostic tool is set up, all the available parameters of the SingleJob are listed in the **SingleJob** configuration dialog, where you can customize them.

The SingleJob data objects must be created in the Project Manager as project-specific parameters. The SingleJob data object of automation blocks must be parameterized by referencing these project-specific data objects.

**Preconditions**

- You have created the required data objects in the Project Manager, refer to How to Set Up Diagnostic Projects on page 13.
- AutomationDesk is connected to the diagnostic tool.

**Method**

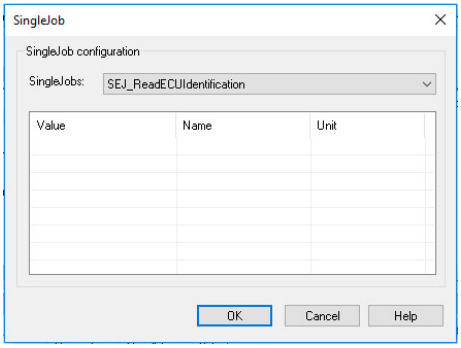
**To configure the SingleJob data object**

1 Add a **SingleJob** data object to the **SingleJobs** data container in the Project Manager. You can do this in two different ways:
   - Open the **LogicalLink** configuration dialog and drag the required SingleJob to the **SingleJobs** data container. The created **SingleJob** data object is

parameterized with the values predefined in the diagnostic project and given the name of the SingleJob.

- Choose **Add SingleJob** from the **SingleJobs** data container's context menu to add a **SingleJob** data object. Double-click the **SingleJob** data object to open the **SingleJob** configuration dialog. Choose a SingleJob from the drop-down list.

**2** Modify the values of the specified **SingleJob** data object. Open the **SingleJob** configuration dialog by double-clicking the SingleJob data object. If the SingleJob provides parameters, they are displayed with their values, names, and units. If there are predefined value ranges specified in the project's database, you can modify a value using the context menu of the parameter entry (right-click the value), otherwise you can enter any value.




**Tip**

Values which must be set in hexadecimal notation are displayed as `0x00`. Several successive hexadecimal entries have to be separated by commas. For example, `0xa1,f2,44`.

**3** Click **OK** to confirm your settings and close the dialog.

**Result**

You have specified a SingleJob data object that you can use as a data object of an automation block.

**Next step**

After you have specified all ControlPrimitive, Service, and SingleJob data objects, you can create the automation sequence, refer to How to Build a Basic Sequence For Diagnostic Tasks on page 22.

**Related topics**

Basics



HowTos



Examples



References



# How to Build a Basic Sequence For Diagnostic Tasks

**Objective**

Each sequence that you build for a diagnostic task contains common blocks. This basic sequence realizes the connection between AutomationDesk and the diagnostic tool, and communication between the diagnostic tool and the ECU.

**Preconditions**

- You have created the required data objects in the Project Manager, refer to How to Set Up Diagnostic Projects on page 13.
- As the diagnostic tool, ControlDesk is installed and configured on your host PC.

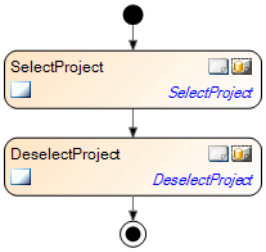
**Method**

**To build a basic sequence for a diagnostic task**

1. Choose **New Sequence** from the AutomationDesk project element's context menu to add a new sequence element to the Project Manager.
2. Double-click the sequence element to open it in the Sequence Builder.
3. Drag a **SelectProject** block from the Library Browser to the Sequence Builder to realize the start of diagnostic project handling.

   The **SelectProject** block considers the states of the System and Project data objects. If the system is already connected and the project is selected, this block has no task to be executed. Otherwise, it connects to the system and selects the project.

4 Drag a **DeselectProject** block from the Library Browser to the Sequence Builder to realize the end of diagnostic project handling.

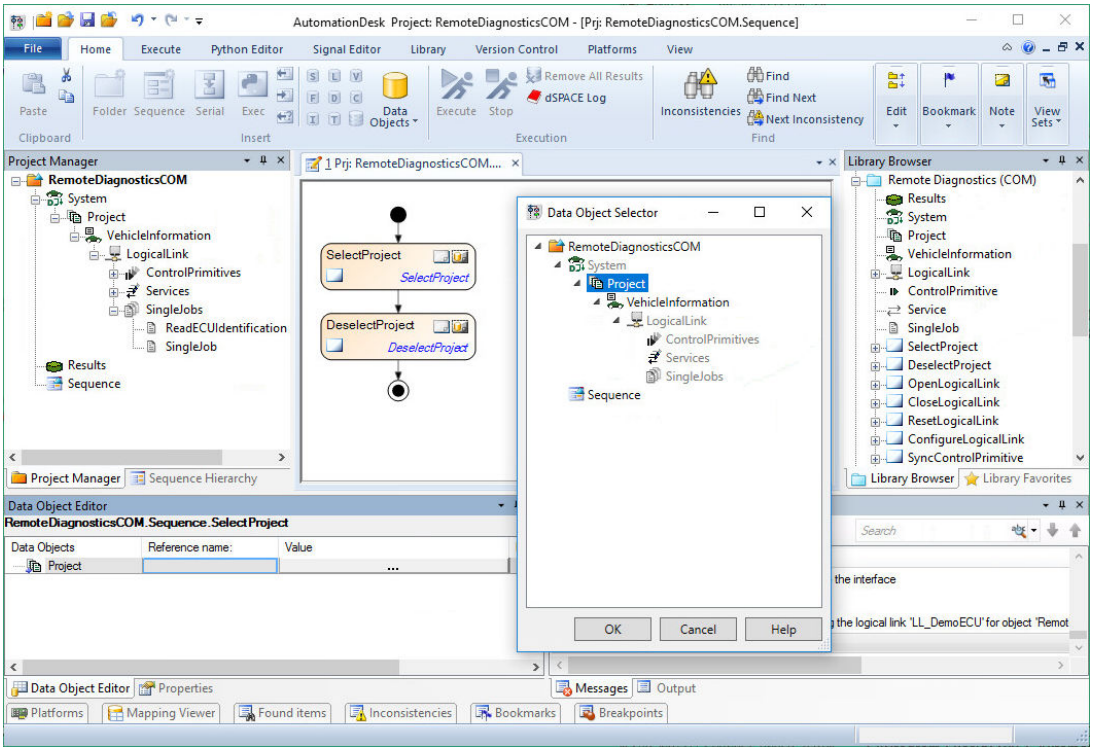



**Tip**

The **DeselectProject** block deselects the project and disconnects the diagnostic system.
You can use the block in different use cases:

- If you want to switch to another diagnostic project during the execution of your sequence.
- If you execute a diagnostic project with several sequences it is sufficient to set the **DeselectProject** block in the very last sequence.

5 Via the **Data Object Editor**, use the **Data Object Selector** to set the project-specific **Project** data object as a reference to these two blocks for the Project data objects.




**Result**

When you execute this sequence, AutomationDesk is connected to the diagnostic tool and loads all information from the ODX project data. Then communication between diagnostic tool and the ECU is started. The project-specific Results data object is set to a value representing the communication state. Then communication stops, the diagnostic project is deselected and the diagnostic tool is disconnected from AutomationDesk, if there is a DeselectProject block.

**Note**

When execution finishes, the Results data object is reset to *None*.

**Next step**

This sequence provides all the information that is required to enlarge it for different use cases. For synchronous access to the diagnostic tool, you can proceed with How to Read Data From the Diagnostic Tool Synchronously on page 25.

**Related topics**

HowTos



Examples



References



# How to Read Data From the Diagnostic Tool Synchronously

**Synchronous read method**

A synchronous service is finished when the diagnostic results are returned. The following service must wait until then.

**Preconditions**

- You have created the required data objects in the Project Manager, refer to How to Set Up Diagnostic Projects on page 13.
- You have created the basic sequence, refer to How to Build a Basic Sequence For Diagnostic Tasks on page 22.
- As the diagnostic tool, ControlDesk is installed on your host PC.

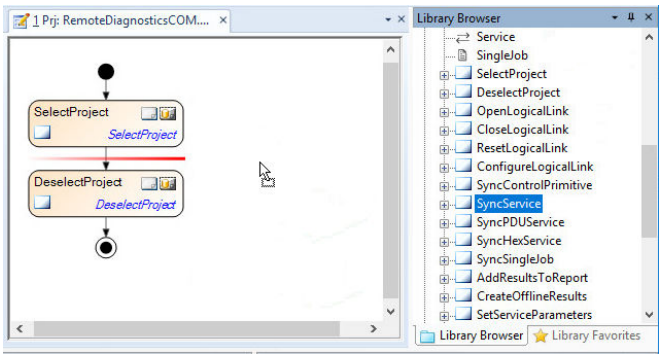
**Note**

The following instruction describes how to use SyncService blocks, but also applies to using SyncControlPrimitive, and SyncSingleJob blocks.

**Method**

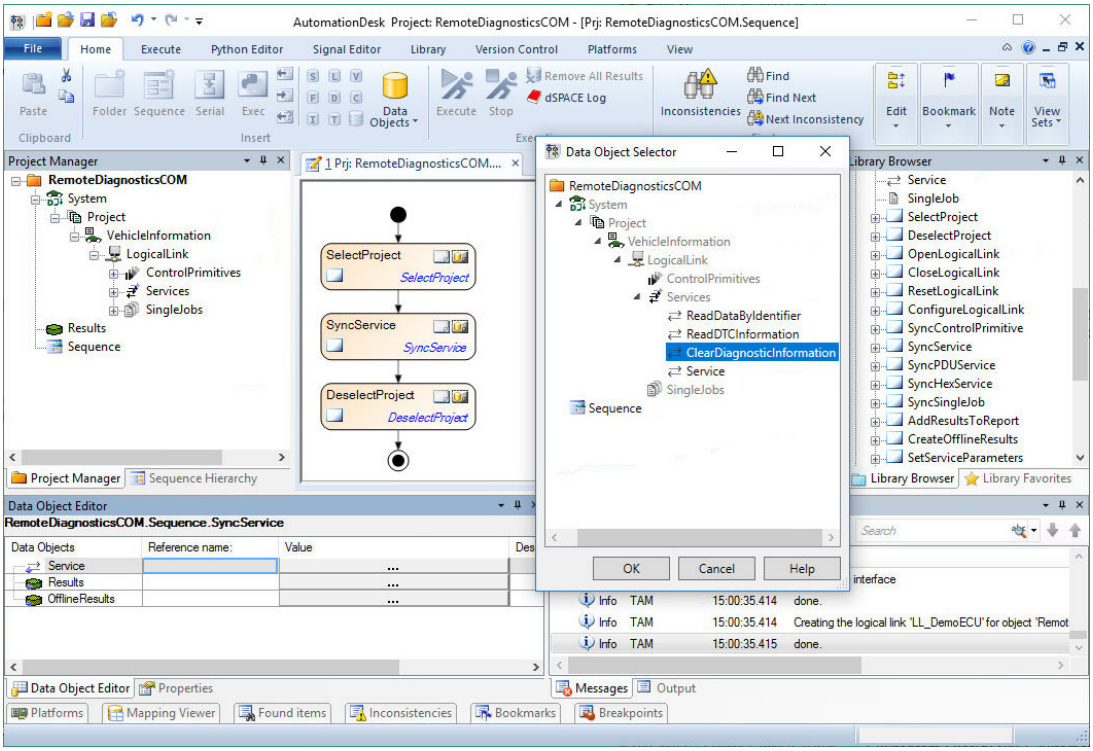
**To read data from the diagnostic tool synchronously**

1 Drag a SyncService block between the SelectProject and DeselectProject blocks.




2 Add a Service data object to the Services data container in the Project Manager and parameterize it with the service you want to automate. For instructions on doing this, refer to How to Configure Service Data Objects on page 18.

3 Via the Data Object Editor, use the Data Object Selector to set the project-specific Results data object and the corresponding Service data object as references to the data objects of the SyncService block.




**Result**

When you execute this sequence, AutomationDesk activates the diagnostic tool to execute the specified diagnostic service. The result of the service is returned and stored in the Results data object.

**Note**

When execution finishes, the Results data object is reset to *None*.

**Next step**

If you want to store the diagnostic results permanently, you can add the temporarily available diagnostic results to an AutomationDesk report. For information on how to do this, refer to How to Add the Results to a Report on page 34.

**Related topics**

Basics



HowTos

Examples



References



# How to Read Data of Low-Level Diagnostic Tasks

**Read method using raw data**

For low-level diagnostic tasks, you can also use raw data. This means that you specify directly the request protocol data unit (PDU) that can be handled by the diagnostic tool, instead of specifying a symbolic data that must be transformed according to the ODX database used.

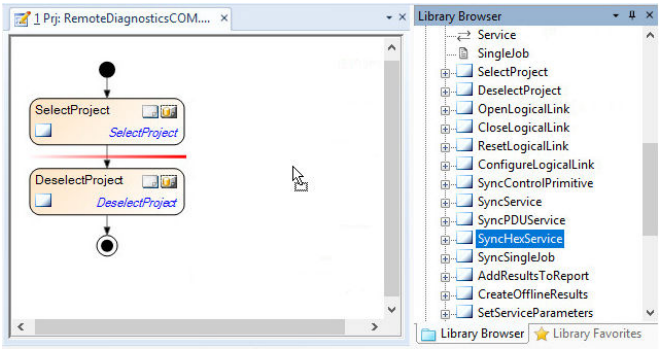
**Preconditions**

- You have created the required data objects in the Project Manager, refer to How to Set Up Diagnostic Projects on page 13.
- You have created the basic sequence, refer to How to Build a Basic Sequence For Diagnostic Tasks on page 22.
- As the diagnostic tool, ControlDesk is installed on your host PC.

**Method**

**To read data of low-level diagnostic tasks**

1 Drag a SyncHexService block between the SelectProject and DeselectProject blocks.



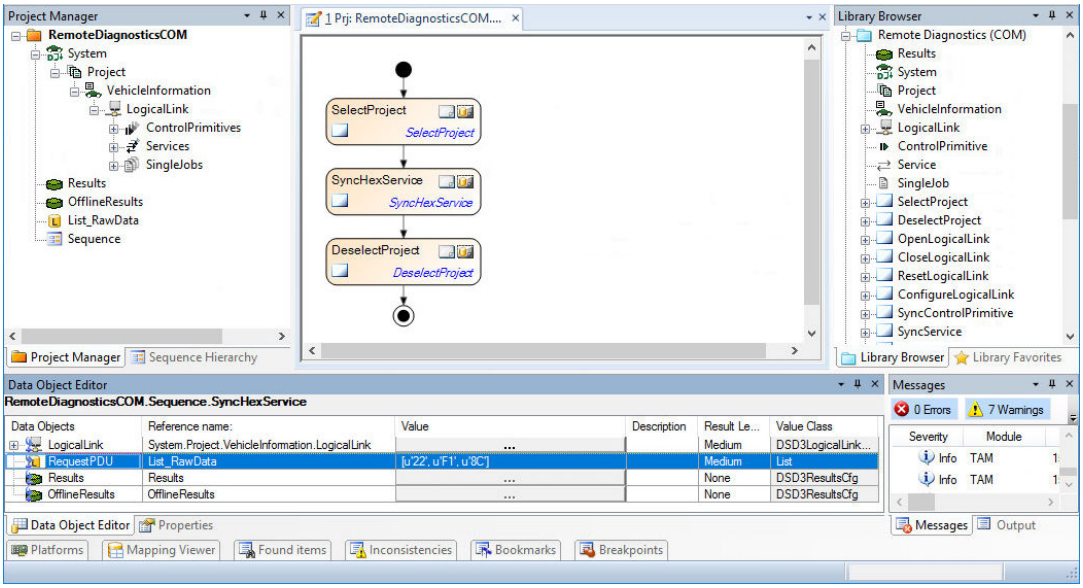

2 Add a List data object to the Project Manager and parameterize it with the values of the service you want to automate. This contains the values of the request PDU. You can specify the hexadecimal values as integer values or as strings.

For example, to specify the request PDU `0x22,F1,8C`, you can create a list containing the hexadecimal values as unicode strings [u'22',u'F1',u'8C'] or as the related integer values [34,241,140].

**Note**

Do not mix strings and integer values in one list.

3 Via the Data Object Editor, use the Data Object Selector to set the project-specific LogicalLink data object, List data object, Results data object, and optionally the OfflineResults data object as references to the data objects of the SyncHexService block.




**Result**

When you execute this sequence, AutomationDesk activates the diagnostic tool to execute the specified diagnostic service. Instead of specifying a Service data object that is implicitly transformed from the symbolic data to the corresponding request PDU, the raw data is directly used to request the diagnostic service. The result of the service is returned and stored in the Results data object.

**Note**

When execution finishes, the Results data object is reset to *None*.

**Next step**

If you want to make the diagnostic results visible, you can add the temporarily available diagnostic results to an AutomationDesk report. For information on how to do this, refer to How to Add the Results to a Report on page 34.

If you want to make the diagnostic results available in offline execution mode, you can store the results as a Python object to the AutomationDesk project. For information on how to do this, refer to How to Get Diagnostic Results For Offline Execution on page 31.

**Related topics**

Basics



HowTos

Examples



References



# How to Get Diagnostic Results For Offline Execution

**Handling the Results data object**

In online execution mode, the contents of a Results data object is only available during the execution of the project. After execution, the value of a Results data object is reset to *None*. While a result exists, you can add it to a report to make it visible also after execution. But, if you want to run a diagnostic task in offline execution mode, you must parameterize an automation block's OfflineResults data object with a Results data object that provides an equivalent data structure. To get such a data object, you can use the **CreateOfflineResults** automation block.

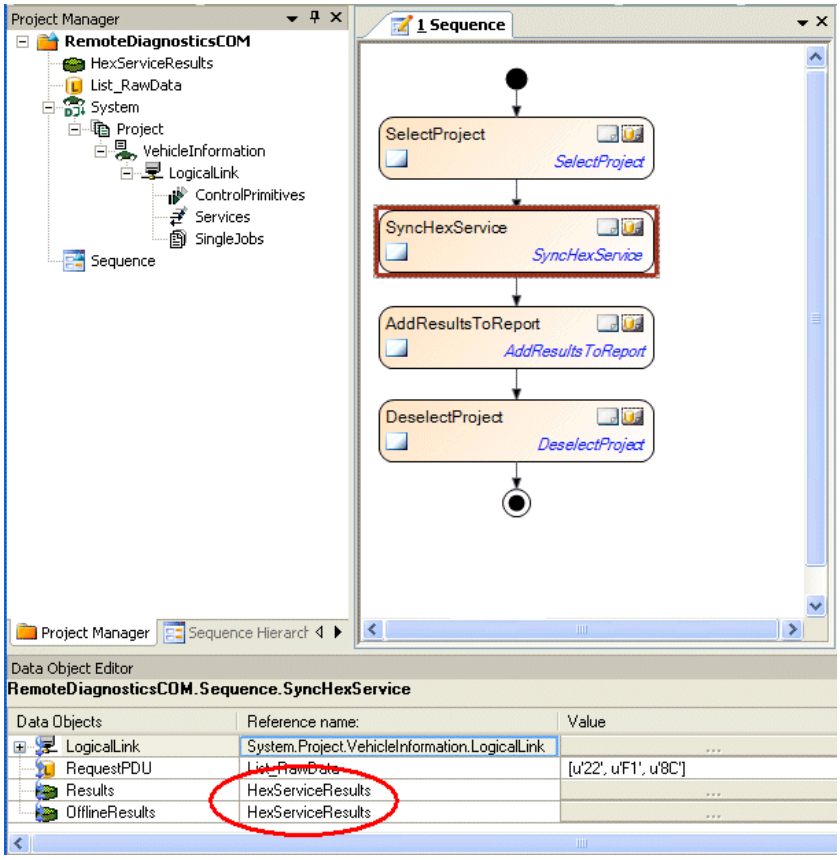
**Preconditions**

- You have created the required data objects in the Project Manager, refer to How to Set Up Diagnostic Projects on page 13.
- You have created the basic sequence, refer to How to Build a Basic Sequence For Diagnostic Tasks on page 22.
- As the diagnostic tool, ControlDesk is installed on your host PC.

**Method**

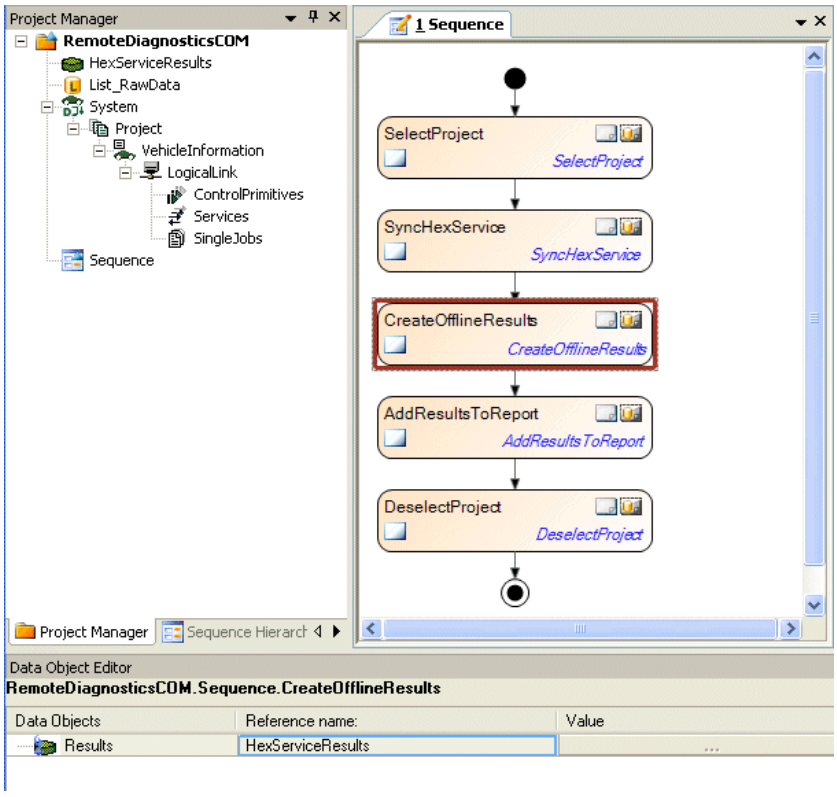
**To get diagnostic results for offline execution**

1. Parameterize the **OfflineResults** data object of a read automation block (SyncControlPrimitive, SyncService, SyncHexService or SyncSingleJob) with the same **Results** data object that is used for the block's **Results** data object.




**2** Drag a **CreateOfflineResults** block after the read block (SyncControlPrimitive, SyncService, SyncHexService or SyncSingleJob).

3 Parameterize the **CreateOfflineResults** automation block with the read block's **Results** data object.




**Result**

When you execute this sequence in online execution mode, the read automation block returns the Results data object. The CreateOfflineResults data object transform the results into a Python object that is stored in the AutomationDesk project. If you then execute the sequence in offline execution mode, the read automation block uses the transformed Results data object as input. The CreateOfflineResults automation block is not executed in offline execution mode.

**Tip**

You can use the results of the CreateOfflineResults automation block in online execution mode too.

**Next step**

If you want to make the diagnostic results visible, you can add the diagnostic results to an AutomationDesk report. For information on how to do this, refer to How to Add the Results to a Report on page 34.

**Related topics**

Basics

Executing Sequences Using Different Operation Modes (AutomationDesk Basic Practices 🕮)
Overview of the Remote Diagnostics (COM) Library..... 8

HowTos

How to Build a Basic Sequence For Diagnostic Tasks..... 22
How to Set Up Diagnostic Projects..... 13

Examples

Example of a Diagnostic Project Sequence..... 10

References

CreateOfflineResults..... 44

# How to Add the Results to a Report

**Objective**

The diagnostic results are only available while the sequence is running. If the diagnostic task has finished, the Results data object is reset to *None*. To store the results permanently, you can add them to an AutomationDesk report using an AddResultsToReport block. The contents of the report depend on the results which are provided by the specified diagnostic services, and single ECU jobs.

**Note**

If you use only one project-specific Results data object that is referenced from all automation blocks, it will be overwritten each time. You must therefore add the results to the report before another automation block returns its value.

**Restrictions**

It is not possible to execute an AddResultsToReport block as the last block in a sequence if a DeselectProject block is executed beforehand.

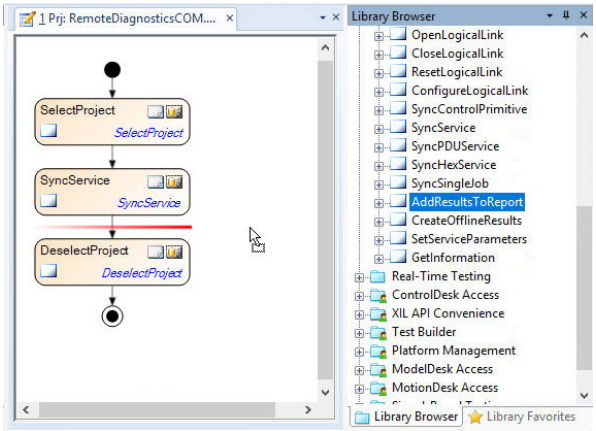
**Precondition**

- You have created the basic sequence, refer to How to Build a Basic Sequence For Diagnostic Tasks on page 22.
- As the diagnostic tool, ControlDesk is installed on your host PC.

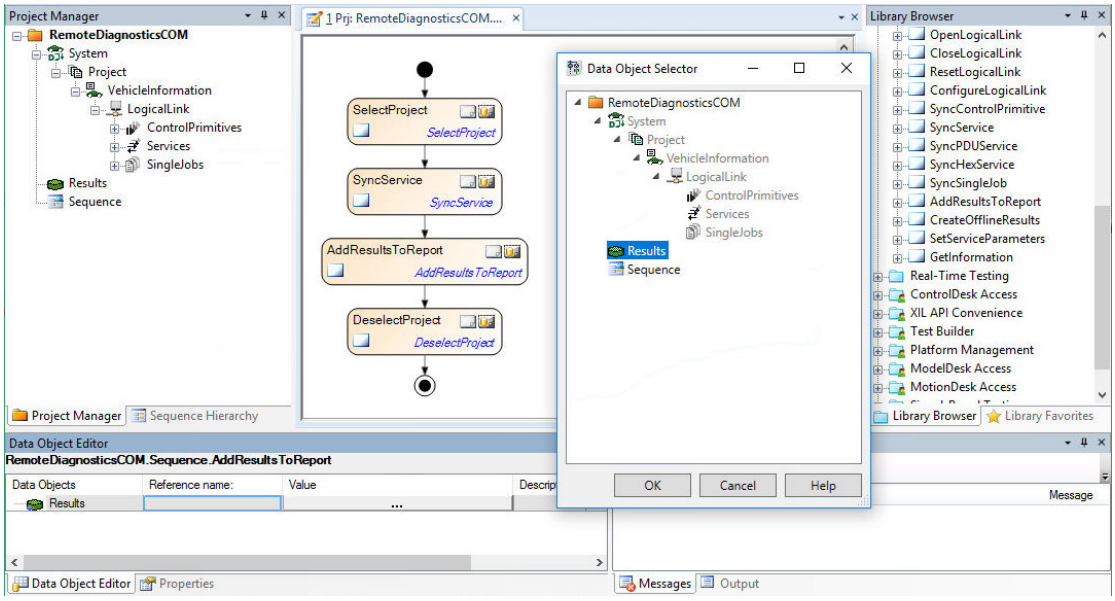
**Method**

**To add the results to a report**

1 Drag the AddResultsToReport block after the block that provides the results which you want to store in the report.




2 Via the Data Object Editor, use the Data Object Selector to set the project-specific Results data object (here: Project_Results) as a reference to the Results data object of the AddResultsToReport block.




**Result**

If you execute the sequence, the currently stored diagnostic results are delivered to the AddResultsToReport block. It generates a table in the report containing service-specific diagnostic values.

## Related topics

Basics

Generating Reports (AutomationDesk Basic Practices 🕮)

HowTos

How to Build a Basic Sequence For Diagnostic Tasks..... 22

Examples

Example of a Diagnostic Project Sequence..... 10

References

AddResultsToReport..... 39

# Reference Information

**Where to go from here**

Information in this section

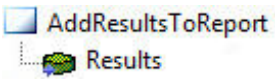
# Automation Blocks

**Where to go from here**

Information in this section

# AddResultsToReport

**Graphical representation**




**Purpose**

To add the results to a report.

**Description**

The AddResultsToReport block allows you to add the results of the collected signals from the diagnostic tool to a report. The results of the SyncControlPrimitive, SyncService, and SyncSingleJob blocks are volatile and available only during the execution of the project. If you want the result of one of these blocks to be added to the report of your project, you have to drag an AddResultsToReport block under the block concerned. The Results data object of both blocks must be referenced to the Results data object of your project via the Data Object selector.

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| Results | In | Results | None | Contains the referenced results. |

**Related topics**

HowTos

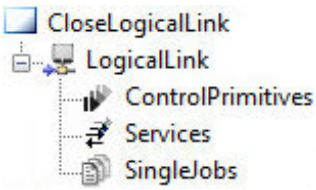
How to Add the Results to a Report..... 34

References

Results..... 50

# CloseLogicalLink

**Graphical representation**




**Purpose**

To disconnect the referenced logical link from the ECU.

**Description**

The CloseLogicalLink automation block is used to terminate the communication with the connected ECU. For example, if the ECU must be powered down during the test, you must close the ECU communication beforehand. If a diagnostic task (control primitive, service or single job) is executing when this block is called, an exception is returned. To close a logical link even if a diagnostic task is running, you must use the ResetLogicalLink automation block. If a logical link is closed, you cannot execute the provided control primitives, services and jobs. If the specified logical link is already closed when this block is called, no action is performed.

To reconnect the logical link with the ECU, you can use the OpenLogicalLink automation block.

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| LogicalLink | In | LogicalLink | None | Defines the logical link that you want to disconnect from the ECU. |

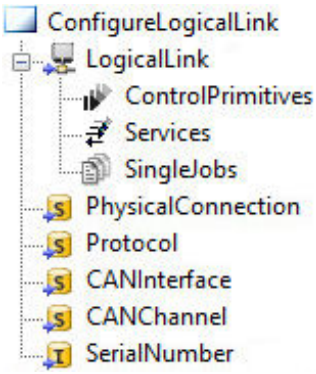
**Related topics**

References



# ConfigureLogicalLink

**Graphical representation**




**Purpose**

To configure the referenced logical link to the ECU during run time.

**Description**

The ConfigureLogicalLink automation block is used to configure/reconfigure a logical link during run time.

**Note**

The ConfigureLogicalLink automation block is functional only if the AutomationDesk Remote Diagnostics (COM) library is in online operation mode and ControlDesk's diagnostic interface `ControlDesk.DSystem` is used.

The diagnostic project must be selected before you can use the ConfigureLogicalLink block to get the expected functionality. So, the ConfigureLogicalLink block must be located after a SelectProject block in an automation sequence. It is recommended to place the ConfigureLogicalLink block in your sequence directly after the SelectProject block.

You can configure the logical link as long as the diagnostic project is selected. The logical link is in the state eCREATED during configuration.

If the settings of the ConfigureLogicalLink block are not valid, an exception is raised.

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| LogicalLink | In | LogicalLink | None | Defines the logical link to which you want to configure. |
| PhysicalConnection | In | String | "Simulation" | Lets you specify the interface module for a logical link.<br>The valid values are:<br>▪ CAN<br>▪ K-Line<br>▪ Simulation |
| Protocol | In | String | "None (Simulation)" | Lets you specify the diagnostic protocol to be used with the logical link.<br>The valid values are:<br>▪ "ISO 14229 (UDS)"<br>▪ "ISO 15765 (Diagnostics on CAN)"<br>▪ "ISO 14230 (KWP2000 on K-Line)"<br>▪ "GMLAN"<br>▪ "TP 2.0"<br>▪ "OBD"<br>▪ "None (Simulation)" |
| CANInterface | In | String | "dSPACE DCI-CAN2" | Lets you specify the physical CAN interface to be assigned to the logical link.<br>The valid values are:<br>▪ "dSPACE DCI-CAN2"<br>▪ "dSPACE Calibration Hub"<br>▪ "dSPACE Virtual CAN channel" |
| CANChannel | In | String | "CAN1" | Lets you specify the CAN controller to be used.<br>The valid values are:<br>▪ "CAN1"<br>▪ "CAN2"<br>(only for Calibration Hub) |
| SerialNumber | In | Int | 0 | Lets you specify the serial number of the physical CAN interface.<br>If 0 is specified as the serial number, CAN interface assignment is automatic. That is, if only one CAN interface is connected, this interface is assigned regardless of its serial number. If several CAN interfaces modules are connected, the physical interface with the lowest serial number is assigned. |

**Related topics**

HowTos



References



# ControlPrimitive

**Graphical representation**


**Purpose**

To provide the basic commands for communication with the ECU.

**Description**

The ControlPrimitive data object describes the diagnostic parameters for communication with the ECU. To get access to the values listed in the ControlPrimitive Configuration dialog you have to do the following beforehand:

1. Connect the diagnostic system
2. Select the diagnostic project
3. Check the vehicle information settings and the selected logical link name

**Note**

When using DCI-Kline1, the port to be used must be set to COM1 in the Windows Device Manager.

**Related topics**

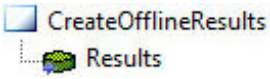
HowTos



References

# CreateOfflineResults

**Graphical representation**




**Purpose**

To create a storable diagnostic result in online mode.

**Description**

The contents of a Results data object are volatile. You can access them only during sequence execution while the corresponding COM object of the diagnostic result is available. If you want to save a measured result in your AutomationDesk project, you must use the CreateOfflineResults block to convert it to a storable Python data structure. If such a result is included in your AutomationDesk project, you can use it as an offline result. The structure of the data object is based upon the ASAM MCD3-D standard.

If you execute blocks of the Remote Diagnostics (COM) library that provide an OfflineResults data object in online recording mode, the referenced project-specific Results data object will also contain the Python data structure.

The block is not executed:

- In offline execution mode.
- If the project is deselected (see DeselectProject on page 45).

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| Results | In / Out | Results | None | Contains the referenced results. After execution of the block, the referenced Results data object contains a storable Python data structure. |

**Related topics**

Basics

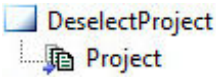
Basics of Operation Modes (AutomationDesk Basic Practices 🕮)

References

Results..... 50

# DeselectProject

**Graphical representation**




**Purpose**

To deselect the currently loaded project.

**Description**

The DeselectProject block deselects the currently loaded project and disconnects from the diagnostic tool.

**Data objects**

This automation block provides the following data object:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| Project | In | Project | None | States the name of the project to be deselected. |

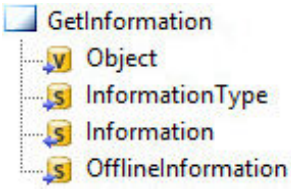
**Related topics**

References

Edit (Project)..... 73
Project..... 48
SelectProject..... 50

# GetInformation

**Graphical representation**




**Purpose**

To get information from the referenced data object.

**Description**

The information on the referenced data object is saved to the Information data object. You can reference the following data objects via the Object parameter value:

- System
- Project
- VehicleInformation
- LogicalLink
- ControlPrimitive
- Service
- SingleJob

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| InformationType | In | String | " " | Defines the type of information you want to know:<br>▪ Description<br>▪ ShortName<br>▪ LongName |
| Object | In | Variant | None | Defines the data object for which you want to get the information. |
| Information | Out | Variant | None | Contains the information. |
| OfflineInformation | Out | Variant | None | Lets you specify the values to be used in offline operation mode. |

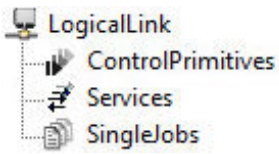
**Related topics**

References

ControlPrimitive.....43
Service.....51
SingleJob.....53

# LogicalLink

**Graphical representation**



**Purpose**

To set up the logical links to the ECU.

**Description**

The LogicalLink data object holds containers for ControlPrimitives, Services, and SingleJobs to set up the connection to the ECU. If you want to drag and drop a ControlPrimitive, Service, or SingleJob from the LogicalLink Configuration dialog in the corresponding container of your project you have to do the following beforehand:

1. Connect the diagnostic system
2. Select the diagnostic project
3. Check the vehicle information settings and the selected logical link name

For the logical link, you can choose not only the base ECU but also ECU variants.

**Related topics**

References



# OpenLogicalLink

**Graphical representation**


**Purpose**

To connect the referenced logical link to the ECU.

**Description**

The OpenLogicalLink automation block is used to reopen and switch online a logical link that has been closed by using the CloseLogicalLink or ResetLogicalLink automation block. If the referenced logical link is already opened, no action is performed.

It is not possible to use this block for switching between different logical links, for example between the base and ECU variants of a logical link, while the diagnostic project is selected.

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| LogicalLink | In | LogicalLink | None | Defines the logical link to which you want to connect the ECU. |

**Related topics**

References

CloseLogicalLink.....40
LogicalLink.....46
ResetLogicalLink.....49

# Project

**Graphical representation**


**Purpose**

To specify a diagnostic project.

**Description**

The Project data object is used to specify the diagnostic project to be automated. If you execute the SelectProject automation block or if you select the project in the Project Configuration dialog, the ODX database of the specified project is loaded to AutomationDesk.

**Note**

- If you use ControlDesk.DSystem202 as the ASAM MCD-D3 interface, the ControlDesk project root folder on the ControlDesk server must contain the ControlDesk projects that include the ECU Diagnostics devices related to the ODX databases you want to use. The name of the ControlDesk experiment that contains the ECU Diagnostics device matches the name of the diagnostic project.
  To display the ControlDesk project root folder list and change the position of root folders, start ControlDesk and open the **Project** page of the **ControlDesk Options** dialog. For details, refer to Project Page (ControlDesk Project and Experiment Management 🕮).
- You cannot switch from one project to another during run time.
- During an automation session involving diagnostic tasks, do not load a ControlDesk experiment containing an ECU Diagnostics device.

If an error occurs during the selection of a project, AutomationDesk disconnects from ControlDesk.

**Related topics**

HowTos

How to Set Up Diagnostic Projects..... 13

References

Edit (Project)..... 73

# ResetLogicalLink

**Graphical representation**


**Purpose**

To disconnect the referenced logical link from the ECU even if a diagnostic task is running.

**Description**

The ResetLogicalLink automation block is used to terminate the communication with the connected ECU, even if a diagnostic task (control primitive, service or single job) is running. If you want to prevent disconnection from the ECU while a diagnostic task is running, you must use the CloseLogicalLink automation block. If a logical link is disconnected, you cannot execute the provided control primitives, services and jobs. If the referenced logical link is already closed, no action is performed.

To reconnect the logical link with the ECU, you can use the OpenLogicalLink automation block.

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| LogicalLink | In | LogicalLink | None | Defines the logical link you want to disconnect from the ECU. |

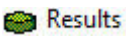
**Related topics**

References



# Results

**Graphical representation**




**Purpose**

To manage the results of the ECU diagnostics.

**Description**

The Results data object contains the referenced results of the SynControlPrimitive, SyncService, and SynSingleJob blocks during the execution of the project. You can save the results to the report permanently by using the AddResultsToReport automation block (refer to AddResultsToReport on page 39).

An OfflineResult data object is required when executing a diagnostic task in offline operation mode. For an example, how to generate an offline diagnostic result, look at the sequence in the **CreateOfflineResult** folder in the **ControlDesk D3System202** demo project.

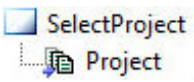
**Related topics**

References



# SelectProject

**Graphical representation**




**Purpose**

To select and load a diagnostic project.

**Description**

When you have selected a project, the SelectProject block sets up a connection to the diagnostic tool and loads the project. All project data is loaded automatically and can be used by the automation blocks. The SelectProject block must always reference a valid and parameterized Project data object.

**Note**

- If you use ControlDesk.DSystem202 as the ASAM MCD-D3 interface, the ControlDesk project root folder on the ControlDesk server must contain the ControlDesk projects that include the ECU Diagnostics devices related to the ODX databases you want to use. The name of the ControlDesk experiment that contains the ECU Diagnostics device matches the name of the diagnostic project.
  To display the ControlDesk project root folder list and change the position of root folders, start ControlDesk and open the **Project** page of the **ControlDesk Options** dialog. For details, refer to Project Page (ControlDesk Project and Experiment Management 🕮).
- You cannot switch from one project to another during run time.
- During an automation session involving diagnostic tasks, do not load a ControlDesk experiment containing an ECU Diagnostics device.

If an error occurs during the selection of a project, AutomationDesk disconnects from ControlDesk.

**Data objects**

This automation block provides the following data object:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| Project | In | Project | None | Defines the name of the project to be loaded. |

**Related topics**

HowTos

How to Set Up Diagnostic Projects ..... 13

References

DeselectProject ..... 45
Edit (Project) ..... 73
Project ..... 48

# Service

**Graphical representation**

**Purpose**

To provide a diagnostic service.

**Description**

When you set up a connection to the diagnostic tool, all the parameters of the available services are listed. The values of the parameters can be customized. All services are provided, if the system is connected properly, a valid project was selected, and the vehicle information and logical link are parameterized.

**Related topics**

HowTos

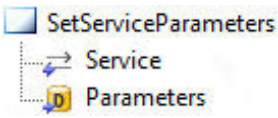


References



## SetServiceParameters

**Graphical representation**




**Purpose**

To set service parameters during execution.

**Description**

The SetServiceParameters automation block is used to modify the parameters of a service. You can specify the parameters and their values using a Dictionary data object. Sometimes there are dependencies between specified values and the number of accessible parameters. If your dictionary contains parameters that are currently not available in the parameter list, they are ignored and a warning message is logged. If a parameter is added to the parameter list dynamically but is not in your dictionary, its default will be used.

**Note**

- Working with this block requires good knowledge about the ODX database.
- Because the dictionary must be filled manually, you should check the correctness of the entered parameter names and values before starting an execution.

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| Service | In | Service | None | Defines the service for which you want to modify the parameters. |
| Parameters | In | Dictionary | {} | Defines the parameters and their values in the form: {u'ParameterShortName':Value}<br>You have to enter a byte field as string, separated by commas or blanks. Examples:<br>▪ Using comma as separator: {u'PA_DataRecord':u'11,22,33'}<br>▪ Using blank as separator: {u'PA_DataRecord':u'11 22 33'} |

**Related topics**

References

Service.....51

# SingleJob

**Graphical representation**


**Purpose**

To provide a single ECU job.

**Description**

The SingleJob data object describes diagnostic parameters for the communication with one ECU (single ECU job). To get access to the values listed in the SingleJob Configuration dialog you have to do the following beforehand:

1. Connect the diagnostic system
2. Select the diagnostic project

3. Check the vehicle information settings and the selected logical link name

**Related topics**

HowTos

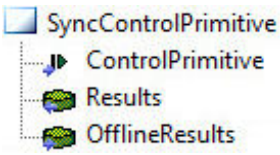
How to Configure SingleJob Data Objects..... 20

References

Add SingleJob..... 66
AddResultsToReport..... 39
Edit (SingleJob)..... 75

# SyncControlPrimitive

**Graphical representation**




**Purpose**

To execute the selected ControlPrimitive data object synchronously.

**Description**

If the ControlPrimitive block is executed it returns a result. The result is saved to the global Results data object if you reference the Results data object of the ControlPrimitive block to it. The results of the ControlPrimitive are volatile. Use the AddResultsToReport block to save the results to a report permanently (see AddResultsToReport on page 39).

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| ControlPrimitive | In | ControlPrimitive | None | Defines the name of the selected ControlPrimitive data object. |
| Results | Out | Results | None | Contains the results of the SyncControlPrimitive data object. |
| OfflineResults | Out | Results | None | Lets you specify the reference to a project-specific Results data object. In online recording mode, the contents of the block's Results data object is converted to a Python data structure and stored in the referenced Results data object. |

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| | | | | In offline mode, the referenced Results data object is used as the block's Results data object. |

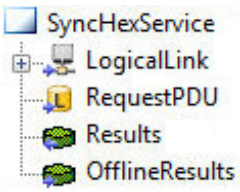
**Related topics**

References

Add ControlPrimitive.....62
Edit (ControlPrimitive).....70
SyncService.....58

# SyncHexService

**Graphical representation**




**Purpose**

To synchronously execute services using raw data instead of symbolic data.

**Description**

The SyncHexService must be used when you want to parameterize the Protocol Data Unit (PDU) of a service request directly with hexadecimal values instead of using the related symbolic data from an ODX database. The PDU data set contains the data to be requested in list format. If the SyncHexService block is executed it returns a diagnostic result. The result is stored to the global Results data object if you reference the Results data object of the SyncHexService block to it. The results of the SyncHexService are volatile. Use the AddResultsToReport block to save the results to a report permanently (see AddResultsToReport on page 39).

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| LogicalLink | In | LogicalLink | None | Lets you specify the LogicalLink to be used. |
| RequestPDU | In | List | [ ] | Lets you specify a list of integer values in the range 0x00 … 0xFF, representing the bytes of the PDU. The elements of the list can also be specified as strings. |

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| | | | | Example: RequestPDU=[u"1A", u"90"]<br>The elements of the list must be of the same data type. |
| Results | Out | Results | None | Returns the results of the specified service. |
| OfflineResults | Out | Results | None | Lets you specify the reference to a project-specific Results data object. In online recording mode, the contents of the block's Results data object is converted to a Python data structure and stored in the referenced Results data object. In offline mode, the referenced Results data object is used as the block's Results data object. |

**Related topics**

HowTos

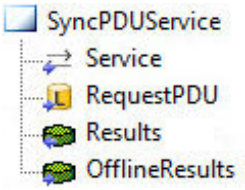
How to Read Data of Low-Level Diagnostic Tasks.....28

References

Add Service.....65
Edit (Service).....74
SyncService.....58

# SyncPDUService

**Graphical representation**




**Purpose**

To synchronously execute services using a RequestPDU.

**Description**

If you use the SyncPDUService, the specified service must already exist, and it is parameterized by the specified RequestPDU. If the SyncPDUService block is executed it returns a diagnostic result. The result is interpreted by using the related symbolic data from an ODX database. It is stored to the global Results data object if you reference the Results data object of the SyncPDUService block to it. The results of the SyncPDUService are volatile. Use the AddResultsToReport block to save the results to a report permanently (see AddResultsToReport on page 39).

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| Service | In | Service | None | Lets you specify the Service to be used. |
| RequestPDU | In | List | [ ] | Lets you specify a list of integer values in the range 0x00 … 0xFF, representing the bytes of the PDU. The elements of the list can also be specified as strings.<br>Example: RequestPDU=[u"1A", u"90"]<br>The elements of the list must be of the same data type. |
| Results | Out | Results | None | Returns the results of the specified service. |
| OfflineResults | Out | Results | None | Lets you specify the reference to a project-specific Results data object. In online recording mode, the contents of the block's Results data object is converted to a Python data structure and stored in the referenced Results data object. In offline mode, the referenced Results data object is used as the block's Results data object. |

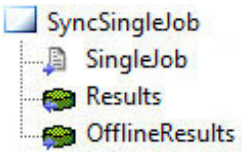
**Related topics**

References

Add Service.....65
Edit (Service).....74
SyncService.....58

# SyncSingleJob

**Graphical representation**




**Purpose**

To execute the selected SingleJob synchronously.

**Description**

If the SyncSingleJob block is executed it returns a result. The result is stored to the global Results data object if you reference the Results data object of the SyncSingleJob block to it. The results of the SyncSingleJob are volatile. Use the AddResultsToReport block to save the results to a report permanently (see AddResultsToReport on page 39).

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| SingleJob | In | SingleJob | None | Defines the name of the selected SingleJob data object. |
| Results | Out | Results | None | Contains the results of the SingleJob data object. |
| OfflineResults | Out | Results | None | Lets you specify the reference to a project-specific Results data object. In online recording mode, the contents of the block's Results data object is converted to a Python data structure and stored in the referenced Results data object. In offline mode, the referenced Results data object is used as the block's Results data object. |

**Related topics**

References

Add SingleJob.....66
Edit (SingleJob).....75
SingleJob.....53

# SyncService

**Graphical representation**


**Purpose**

To execute the selected Service synchronously.

**Description**

If the SyncService block is executed it returns a result. The result is stored to the global Results data object if you reference the Results data object of the SyncService block to it. The results of the SyncService are volatile. Use the AddResultsToReport block to save the results to a report permanently (see AddResultsToReport on page 39).

**Data objects**

This automation block provides the following data objects:

| Name | In / Out | Type | Default Value | Description |
|---|---|---|---|---|
| Service | In | Service | None | Defines the name of the selected Service data object. |
| Results | Out | Results | None | Contains the results of the Service data object. |
| OfflineResults | Out | Results | None | Lets you specify the reference to a project-specific Results data object. In online recording mode, the contents of the block's Results data object is converted to a Python data structure and stored in the referenced Results data object. In offline mode, the referenced Results data object is used as the block's Results data object. |

**Related topics**

HowTos

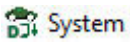
How to Read Data From the Diagnostic Tool Synchronously..... 25

References

Add Service..... 65
Edit (Service)..... 74
SyncControlPrimitive..... 54

# System

**Graphical representation**




**Purpose**

To set the interface and host of the diagnostic tool.

**Description**

You can state the interface name and host name (IP address or computer name) of the diagnostic tool. The interface name and host (IP address or computer name) are valid as long as no other parameters are specified. If a connection to the diagnostic tool was established successfully, all the available projects are displayed.

The interface name is set to **ControlDesk.D3System202** to specify that ControlDesk is used as the diagnostic tool.

**Note**

To avoid incompatibilities between 32-bit versions and 64-bit versions of AutomationDesk and ControlDesk, use the products of the same dSPACE Release.

**Related topics**

HowTos

How to Set Up Diagnostic Projects.....13

References

Edit (System).....76

# VehicleInformation

**Graphical representation**


**Purpose**

To describe the vehicle information.

**Description**

The vehicle information describes which ECUs are installed in the vehicle. The VehicleInformation data object can contain one ore more LogicalLink data objects.

**Related topics**

References

Add VehicleInformation.....67
Edit (VehicleInformation).....77
LogicalLink.....46

# Commands And Dialogs

**Where to go from here**

Information in this section

# Add ControlPrimitive

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ ControlPrimitives data container in the Project Manager<br>▪ ControlPrimitives data container in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To add a communication instruction to the ControlPrimitives data container.

**Result**

A new ControlPrimitive data object is added to the ControlPrimitives data container of a LogicalLink data object.

**Description**

A ControlPrimitive data object must be a subelement of a ControlPrimitives data container. A ControlPrimitives data container can contain multiple ControlPrimitive data objects. You parameterize the ControlPrimitive data object using the **ControlPrimitive Configuration** dialog.

**Note**

If you want to parameterize the ControlPrimitive data object, the diagnostic project must be selected. For further information, refer to Select on page 78.

**Tip**

You can additionally create and use ControlPrimitive data objects at different places in the project hierarchy. These must be referenced to the ControlPrimitive data objects in the diagnostic task definition.
This allows you to change the references, for example, of sequence-specific ControlPrimitive data objects without editing the sequence itself. In some cases it is useful to do so, but it is not required.

**Related topics**

References

ControlPrimitive.....43
Edit (ControlPrimitive).....70
LogicalLink.....46

# Add LogicalLink

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ VehicleInformation data object in the Project Manager<br>▪ VehicleInformation data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To add a LogicalLink data object to a VehicleInformation data object.

**Result**

A new LogicalLink data object is added to a VehicleInformation data object.

**Description**

A LogicalLink data object, which describes the ECU device, must be a subelement of a VehicleInformation data object. A VehicleInformation data object can contain multiple LogicalLink data objects. The settings of a LogicalLink are automatically loaded from the connected diagnostic tool. The LogicalLink can be parameterized using the LogicalLink Configuration dialog, refer to Edit (LogicalLink) on page 72. A LogicalLink data object contains a ControlPrimitives, Services, and SingleJobs data container.

**Note**

If you want to parameterize the LogicalLink data object, the diagnostic project must be selected. For further information, refer to Select on page 78.

**Related topics**

References



# Add Project

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ System data object in the Project Manager<br>▪ System data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To add a project to the System data object.

**Result**

The System data object contains a project element, which represents a diagnostic project.

**Description**

A Project data object must be a subelement of a System data object. A system can contain multiple projects. The project settings can be loaded from the diagnostic tool (which must be connected) using the **`Select`** command. You can select only one project at a time.

**Related topics**

References

# Add Service

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ Services data container in the Project Manager<br>▪ Services data container in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To add a Service data object to the Services data container.

**Result**

A new Service data object is added to the Services data object.

**Description**

A Service data object must be a subelement of a Services data container. A Services data container can contain multiple Service data objects. The settings of a Service can be loaded from the connected project using the Service Configuration dialog.

**Note**

If you want to parameterize the Service data object, the diagnostic project must be selected. For further information, refer to Select on page 78.

**Tip**

You can additionally create and use Service data objects at different places in the project hierarchy. These must be referenced to the Service data objects in the diagnostic task definition.
This allows you to change the references, for example, of sequence-specific Service data objects without editing the sequence itself. In some cases it is useful to do so, but it is not required.

**Related topics**

References

Edit (Service)..... 74
Service..... 51

# Add SingleJob

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ SingleJobs data container in the Project Manager<br>▪ SingleJobs data container in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To add a SingleJob data object to the SingleJobs data container.

**Description**

A SingleJob data object must be a subelement of a SingleJobs data container. A SingleJobs data container can contain multiple SingleJob data objects. The settings of a SingleJob can be loaded from the connected project using the SingleJob Configuration dialog.

**Note**

If you want to parameterize the SingleJob data object, the diagnostic project must be selected. For further information, refer to Select on page 78.

**Tip**

You can additionally create and use SingleJob data objects at different places in the project hierarchy. These must be referenced to the SingleJob data objects in the diagnostic task definition.
This allows you to change the references, for example, of sequence-specific SingleJob data objects without editing the sequence itself. In some cases it is useful to do so, but it is not required.

**Related topics**

References



# Add VehicleInformation

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ Project data object in the Project Manager<br>▪ Project data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To add a VehicleInformation data object to a Project data object.

**Result**

A new VehicleInformation data object is added to the Project data object.

**Description**

A VehicleInformation data object must be a subelement of a Project data object. A Project data object can contain multiple VehicleInformation data objects. The settings of a VehicleInformation can be loaded from the connected diagnostic tool using the VehicleInformation Configuration dialog.

**Related topics**

References

# Connect

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ System data object in the Project Manager<br>▪ System data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To connect the diagnostic server (ControlDesk) with AutomationDesk.

**Description**

The diagnostic tool (ControlDesk) and AutomationDesk exchange diagnostic data via COM/DCOM interface. If the system is connected, all project information from the diagnostic tool database can be loaded to AutomationDesk.

**Note**

The DCOM settings for your computer must be performed by a user with administrator rights using the "dcomcnfg" tool, provided by your Windows operating system.

**System Configuration dialog**

**Interface** Specifies the diagnostic tool that is used.

AutomationDesk's Remote Diagnostics (COM) library supports the use of ControlDesk with an installed ControlDesk ECU Diagnostics Module (ControlDeskNG.D3System202) as the diagnostic tool.

**Host** Lets you enter the IP address of the diagnostic server. The default value 127.0.0.1 (local host) is used, if the diagnostic server and AutomationDesk are installed on the same PC.

**Connect/Disconnect – (Available only if the library is set to online operation mode)** Lets you connect/disconnect AutomationDesk to/from the diagnostic tool.

**Related topics**

References

Disconnect..... 69
System..... 59

# Deselect

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ Project data object in the Project Manager<br>▪ Project data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To deselect a previously selected diagnostic project.

**Result**

The project is deselected and the connection to the diagnostic tool is disconnected.

**Description**

If the diagnostic tool is connected to AutomationDesk, you can choose one of the available diagnostic projects as the project to be tested. If you have selected a project, you can deselect it again using this command.

**Project Configuration dialog**

**Project** Displays the currently selected diagnostic project

**Deselect – (Available only if the library is set to online operation mode)** Lets you deselect the Project data object. It is reset to default values.

**Related topics**

References

Project.....48
Select.....78

# Disconnect

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor |

| | |
|---|---|
| | ▪ System data object in the Project Manager<br>▪ System data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To disconnect the diagnostic server from AutomationDesk.

**Result**

The connection between the diagnostic tool and AutomationDesk is closed.

**Description**

The diagnostic tool and AutomationDesk no longer exchange diagnostic data via COM/DCOM interface. AutomationDesk has no access to data of the diagnostic project.

**System Configuration dialog**

**Interface** Specifies the diagnostic tool that is used.

AutomationDesk's Remote Diagnostics (COM) library supports the use of ControlDesk with an installed ControlDesk ECU Diagnostics Module (ControlDeskNG.D3System202) as the diagnostic tool.

**Host** Lets you enter the IP address of the diagnostic server. The default value 127.0.0.1 (local host) is used, if the diagnostic server and AutomationDesk are installed on the same PC.

**Connect/Disconnect – (Available only if the library is set to online operation mode)** Lets you connect/disconnect AutomationDesk to/from the diagnostic tool.

**Related topics**

References

Connect.....68
System.....59

# Edit (ControlPrimitive)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ ControlPrimitive data object in the Project Manager |

| | |
|---|---|
| | ▪ ControlPrimitive data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To configure the selected ControlPrimitive data object.

**Result**

The properties are assigned to the ControlPrimitive data object.

**ControlPrimitive Configuration dialog**

**ControlPrimitive** Lets you select a ControlPrimitive from the drop-down list. The available communication instructions are defined in the LogicalLink data object. You can edit the value of the listed ControlPrimitive parameters. You can select a predefined parameter value for some parameters from their context menus. For all other values, you must enter your input in the appropriate edit fields.

**Tip**

Values which must be set in hexadecimal notation are displayed as `0x00`. Several successive hexadecimal entries have to be separated by commas. For example, `0xa1,f2,44`.

**Add/Edit/Delete – (Available only if the library is set to offline operation mode)** Lets you create and organize a list of ControlPrimitives.

**Related topics**

References

Add ControlPrimitive.....62
ControlPrimitive.....43
LogicalLink.....46

# Edit (GetInformation)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | None |
| Shortcut key | None |

| | |
|---|---|
| Icon | None |
| Others | Double-click the GetInformation block |

**Purpose**

To configure the information to be returned by the GetInformation automation block.

**Result**

The Information data object of the GetInformation automation block contains the specified information.

**GetInformation Configuration dialog**

**Object type** Lets you select the object type you want to get the information from. The following data objects can be referenced:

- System
- Project
- VehicleInformation
- LogicalLink
- ControlPrimitive
- Service
- SingleJob

**Information** Lets you select the information type you want to get. The following types can be selected:

- Description
- LongName
- ShortName

**Related topics**

References

GetInformation.....45

# Edit (LogicalLink)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ LogicalLink data object in the Project Manager<br>▪ LogicalLink data object in the Sequence Hierarchy Browser |

| | |
|---|---|
| Shortcut key | None |
| Icon | None |

**Purpose**

To configure the selected LogicalLink data object.

**Result**

The properties are assigned to the LogicalLink data object.

**LogicalLink Configuration dialog**

**LogicalLink name** Lets you select the LogicalLink specified in the loaded diagnostic project definition. You can select base variants and ECU variants.

**Variable browser – (Available only if the library is set to online operation mode)** Displays information about the Services, ControlPrimitives, and SingleJobs on the selected device. The structure of the loaded LogicalLink is displayed in the hierarchy tree. Here you can select the Service, ControlPrimitive, and SingleJob data objects of the chosen LogicalLink data object. This information is only displayed if the project was selected beforehand.

Every entry takes immediate effect if you select a new item or close the dialog (by clicking the Close button in the title bar).

**Related topics**

References

Add LogicalLink ..... 63
LogicalLink ..... 46
VehicleInformation ..... 60

# Edit (Project)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ Project data object in the Project Manager<br>▪ Project data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To configure the selected Project data object.

**Result**

The selected diagnostic project is assigned to the Project data object.

**Project Configuration dialog**

**Project** Lets you select a diagnostic project from the drop-down list. The available projects are defined in the loaded diagnostic project.

**Select/Deselect – (Available only if the library is set to online operation mode)** Lets you select/deselect the diagnostic project. When you deselect it, it is reset to default values.

If you use ControlDesk.DSystem202 as the ASAM MCD-D3 interface, you can specify the ODX database to be used for communicating with the ECU by selecting the experiment that contains the related ECU diagnostic device. The ControlDesk experiment name matches the diagnostic project name.

**Related topics**

References

Add Project....64
Deselect....69
Project....48
Select....78
System....59

# Edit (Service)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ Service data object in the Project Manager<br>▪ Service data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To configure the selected Service data object.

**Result**

The properties are assigned to the Service data object.

**Service Configuration dialog**

**Functional Classes** Lets you select a functional class from the drop-down list. The available functional classes are defined in the LogicalLink data object.

**Service** Lets you select a service from the drop-down list. The available services are defined in the **LogicalLink** data object and depend on the selected functional class.

**Parameter Configuration** Displays the parameters of the selected service. You can edit the value of the listed service parameters. You can select a predefined parameter value for some parameters from their context menus. For all other values, you must enter your input in the appropriate edit fields.

**Tip**

Values which must be set in hexadecimal notation are displayed as `0x00`. Several successive hexadecimal entries have to be separated by commas. For example, `0xa1,f2,44`.

**Add/Edit/Delete – (Available only if the library is set to offline operation mode)** Lets you create and organize a list of services.

**Related topics**

References

Add Service.....65
Service.....51

# Edit (SingleJob)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ SingleJob data object in the Project Manager<br>▪ SingleJob data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To configure the selected SingleJob data object.

**Result**

The properties are assigned to the SingleJob data object.

**SingleJob Configuration dialog**

**SingleJobs** Lets you select a SingleJob (single ECU job) from the drop-down list. The available SingleJobs are defined in the LogicalLink data object.

**Parameter Configuration** Displays the parameters of the selected SingleJob. You can edit the value of the listed SingleJob parameters. You can select a predefined parameter value for some parameters from their context menus. For all other values, you must enter your input in the appropriate edit fields.

**Tip**

Values which must be set in hexadecimal notation are displayed as `0x00`. Several successive hexadecimal entries have to be separated by commas. For example, `0xa1,f2,44`.

**Add/Edit/Delete – (Available only if the library is set to offline operation mode)** Lets you create and organize a list of SingleJobs.

**Related topics**

References

Add SingleJob.....66
SingleJob.....53

# Edit (System)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ System data object in the Project Manager<br>▪ System data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To configure the selected system data object.

**Result**

The properties are assigned to the system data object.

**System Configuration dialog**

**Interface** Specifies the diagnostic tool that is used.

AutomationDesk's Remote Diagnostics (COM) library supports the use of ControlDesk with an installed ControlDesk ECU Diagnostics Module (ControlDeskNG.D3System202) as the diagnostic tool.

**Host** Lets you enter the IP address of the diagnostic server. The default value 127.0.0.1 (local host) is used, if the diagnostic server and AutomationDesk are installed on the same PC.

**Connect/Disconnect – (Available only if the library is set to online operation mode)** Lets you connect/disconnect AutomationDesk to/from the diagnostic tool.

**Related topics**

References

Connect.....68
Disconnect.....69
System.....59

# Edit (VehicleInformation)

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ VehicleInformation data object in the Project Manager<br>▪ VehicleInformation data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To configure the selected VehicleInformation data object.

**Result**

The selected VehicleInformation is assigned to the VehicleInformation data object.

**VehicleInformation Configuration dialog**

**VehicleInformation** Lets you select one of the available VehicleInformation data from the loaded diagnostic project.

**Related topics**

References



# Select

**Access**

You can access this command via:

| | |
|---|---|
| Ribbon | None |
| Context menu of | ▪ Data Objects column of the Data Object Editor<br>▪ Project data object in the Project Manager<br>▪ Project data object in the Sequence Hierarchy Browser |
| Shortcut key | None |
| Icon | None |

**Purpose**

To select a diagnostic project.

**Note**

If you use ControlDesk.DSystem202 as the ASAM MCD-D3 interface, the ControlDesk project root folder on the ControlDesk server must contain the ControlDesk projects that include the ECU Diagnostics devices related to the ODX databases you want to use. The name of the ControlDesk experiment that contains the ECU Diagnostics device matches the name of the diagnostic project.
To display the ControlDesk project root folder list and change the position of root folders, start ControlDesk and open the **Project** page of the **ControlDesk Options** dialog. For details, refer to Project Page (ControlDesk Project and Experiment Management 🕮).

**Result**

All information that is stored in the selected diagnostic project is available in AutomationDesk via the Project data object.

**Description**

If the diagnostic tool is connected to AutomationDesk, you can choose one of the available diagnostic projects as the project to be tested. The Project data object in AutomationDesk represents a diagnostic project. The available diagnostic information is hierarchically structured and can be recreated by adding

AutomationDesk's VehicleInformation, LogicalLink, ControlPrimitive, Service, and SingleJob data objects to the Project data object.

**Project Configuration dialog**

**Project** Lets you select one of the available diagnostic projects from a drop-down list.

**Select – (Available only if the library is set to online operation mode)** Lets you confirm your selection.

**Related topics**

References

Deselect.....69
Project.....48

# Automation

## Basics on Automating the Access to Remote Diagnostics COM

**Introduction**

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

**Related information**

The AutomationDesk COM API provides the following objects for configuring the access to a diagnostic system via the Remote Diagnostics COM library:

- D3System (AutomationDesk Automation 🕮)
- D3Project (AutomationDesk Automation 🕮)
- D3VehicleInformation (AutomationDesk Automation 🕮)
- D3LogicalLink (AutomationDesk Automation 🕮)
- D3ControlPrimitive (AutomationDesk Automation 🕮)
- D3Service (AutomationDesk Automation 🕮)
- D3SingleJob (AutomationDesk Automation 🕮)
- D3Results (AutomationDesk Automation 🕮)

For basic information and instructions, refer to Basics and Instructions on page 7.

# Limitations

## Limitations When Using the Remote Diagnostics (COM) Library

| | |
|---|---|
| **Connection failed** | Remote Diagnostics (COM) and MATLAB use Xerces as an XML parser. If the Xerces DLLs used are compiled with different compiler versions, they are incompatible. If you try to connect to the diagnostic system, you will get the error message "The specified procedure could not be found." in AutomationDesk. |
| **Execution of SyncSingleJob failed** | If you are working with restricted user rights, the execution of a SyncSingleJob block fails under the following conditions:<br>▪ The diagnostic project is selected and deselected without executing a job in between, before you select the project again and execute the SyncSingleJob block.<br>▪ The diagnostic project is selected and deselected, and the diagnostic system (i.e. ControlDesk application) is not disconnected before you select the project again. |
| **Wrong IP address for server connection when using Windows 7** | If you start the diagnostic server for the first time, the firewall in Windows 7 asks you to allow the access to *ControlDesk Application*. If you confirm the dialog and enter an IP address different to 127.0.0.0, the server will be connected, but AutomationDesk hangs up during execution. |

## V