



<AutomationML/>

**The Glue for Seamless
Automation Engineering**

**Application Recommendation Extension:
Drive for Automation Project
Configuration**

Document Identifier: ARE APC Drive, V 1.2.0

State: April 2020

©AutomationML consortium

Version 1.2.0, April 2020

Contact: www.automationml.org

Table of contents

Table of contents	3
List of figures	4
List of tables	5
1 Introduction.....	6
1.1 Basics.....	6
1.2 Scope	7
1.3 References.....	8
2 General notes regarding exchange of configuration data for drives	9
2.1 Data exchange workflow.....	10
2.2 Recommended workflow.....	10
2.2.1 ECAD engineering.....	11
2.2.2 PLC engineering.....	11
2.2.3 Drive engineering	12
3 Drive Configuration data structures in AutomationML	13
3.1.1 Drive Configuration data exchange data model.....	13
4 Modelling of Drive Configuration data with AutomationML	15
4.1 RoleClassLibrary.....	15
4.1.1 Drive	16
4.1.2 DriveAssetType	16
4.2 InterfaceClassLibrary	17
4.2.1 DriveAssignment	18
4.2.2 DriveAssetsAssignment	18
Appendix A Appendix A: Automation-ML examples.....	20
A.1 Drive Configuration with infeed, inverter and sensor modules (Example of PLC with S120).....	20
A.2 Drive Configuration with double multi-motor-module.....	21

List of figures

Figure 1 – Data exchange for dimensioning, electrical design, programming and diagnosis tools.....	9
Figure 2 Aspects of drive configurations.....	10
Figure 3 – ECAD engineering of a multi-axis drive system	11
Figure 4 – Hardware configuration of a multi-axis drive system within engineering tool.....	11
Figure 5 – Objects and parameters of the Drive Configuration data exchange and their relation to the elements of Automation Project Configuration	13
Figure 6 – AutomationProjectConfigurationDriveExtensionRoleClassLib in AutomationML Editor view	15
Figure 7 – AutomationProjectConfigurationDrivesExtensionRoleClassLib as XML representation	15
Figure 8 –AutomationProjectConfigurationDriveExtensionInterfaceClassLib in AutomationML Editor view	17
Figure 9 – AutomationProjectConfigurationDriveExtensionInterfaceClassLib as XML representation	17
Figure 10 – Assigning Drive and DriveAssetType	18
Figure 11 – Hardware configuration of a multi-axis drive system within engineering tool as XML representation.....	20
Figure 12 – Hardware configuration of a double-motor-module within engineering tool as XML representation	21

List of tables

Table 1 – Overview of AutomationML parts.....	6
Table 2 – Definition Drive.....	16
Table 3 – Definition DriveAssetType	16
Table 4 – Definition DriveAssignment.....	18
Table 5 – Definition DriveAssetsAssignment.....	18

1 Introduction

A very frequently occurring task within the planning process of production and automation systems is the exchange of automation project configuration information of automation system devices between ECAD and engineering systems. This information includes a wide range of diverse hardware modules and systems, like PLCs, decentral IO-devices and drives. To avoid manual transfer of the engineering data between the participating systems, ECAD and engineering systems need an interface for sharing this information. This information will not only include all necessary hardware modules and interconnections, yet also additional logical and structural information of the system, which are provided at an early stage of the planning process and have to be preserved through the whole planning process. Using this automatic data transfer errors on manual copying data can be avoided. Also, for changes of the system, the reaction time can be reduced to a large extend.

With this application recommendation, we extend the configuration information of /AR APC V 1.2.0/ by the definition of a drive.

A drive or motor controller consists of one or more hardware modules that serves to govern in some predetermined manner the performance of an electric motor. Within our specification we will use a dedicated drive object which acts as a representation of the drive and references all hardware modules which are needed to provide this functionality. We also will use a drive object to control other aspects of a drive configuration, as for example power control or handling of additional input signals which contribute to overall drive behavior. A drive object may also be used to control other aspects of a drive system. This application recommendation describes these workflows in accordance to and with the methods of /AR APC V 1.2.0/.

1.1 Basics

The data exchange format AutomationML which is standardised as per IEC 62714 standard, is a neutral, free, and XML-based data format. It has been developed in order to support the data exchange between engineering tools in a heterogeneous engineering tool landscape.

Due to the different aspects of AutomationML the IEC 62714 consists of different parts.

Table 1 – Overview of AutomationML parts

Part / Document Identifier	Title	Description
Part 1 / WP Arch, V 2.0.0	Architecture and general requirements	This part specifies the general AutomationML architecture, the modelling of the engineering data, classes, instances, relations, references, hierarchies, basic AutomationML libraries and extended AutomationML concepts.
Part 2 / WP Lib V 2.0.0	Role class libraries	This part specifies additional AutomationML libraries.
Whitepaper / WP Comm V 1.0.0	Communication	This Whitepaper describes the modelling of Communication mechanisms in AutomationML
Whitepaper / WP eClass V 1.0.0	AutomationML and eCI@ss integration	This Whitepaper describes the integration of eCI@ss in AutomationML
Best Practice Recommendation / BPR MlingExp V 1.0.0	Multilingual expressions in AutomationML	This Whitepaper describes the handling of different texts for different languages in AutomationML
Best Practice Recommendation / BPR RefDes V 1.0.0	Modelling of Reference Designations	This Whitepaper describes the handling of reference designations following IEC 81346-1:2009-07 within AutomationML

Application Recommendation AutomationProject Configuration / AR APC V 1.2.0	AutomationML and ECAD integration	This application recommendation describes the data exchange between ECAD and PLC systems
---	---	---

Further parts may be added in the future in order to e.g. interconnect further data standards to AutomationML.

1.2 Scope

This application recommendation proposes a modelling method of drive configurations based on automation project configuration data by means of the engineering data format AutomationML. It will describe the recommended use of role and interface classes as well as the recommended structures to be considered within the instance hierarchy of an AutomationML document.

1.3 References

The following documents are referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Extensible Markup Language (XML) 1.0:2004, W3C Recommendation (available at <<http://www.w3.org/TR/2004/REC-xml-20040204/>>)

IEC 62424:2008, Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools

Whitepaper AutomationML Part 1 – AutomationML Architecture, November 2018

Whitepaper AutomationML Part 2 – AutomationML Role Libraries, October 2014

Whitepaper AutomationML – AutomationML Communication, September 2014

Whitepaper AutomationML – AutomationML and eCI@ss Integration, November 2017

Best Practice Recommendation Multilingual expressions in AutomationML, March 2017

Best Practice Recommendation Modelling of Reference Designations, September 2017

Application Recommendation Automation Project Configuration, April 2020

2 General notes regarding exchange of configuration data for drives

Systems containing drives may comprise different aspects other than electrical configuration. For systems like robots or tooling machines the machine concept may start with functional view, which includes also dynamic, mechanical and geometrical aspects.

Tools used to define this machine concept at an early stage of the machine design provide information on the functional behaviour of the intended machine. Starting from this initial functional behaviour the mechanical and electrical design of the system will be derived, as well as the programming aspects in a later stage of system implementation.

Based on the functional requirement of the system or machine, the different electrical components can be derived, which are needed to build the machine configuration. Dynamic and mechanical aspects also provide information for the dimensioning of the electrical components, which can be used to select fitting components during the electrical design of the system or machine to be build.

While the mechanical aspects are not relevant for the electrical design¹, the information concerning the drive is essential for the electrical design as well as for the programming aspects of PLC tools.

While a machine concept design tool, ECAD tools and tools for engineering and PLC programming have different views of the automation system information, they share common information, which must be kept consistent among different tools. This does not only apply to the initial configuration of the system, it also applies to changes during system engineering, refitting the system, as well as simulation and diagnosis of the system.

Other tools may also be included within the data transfer or use the information exchanged, as shown in Figure 1. Nevertheless, within this document focus on ECAD and PLC programming tools.

Figure 1 shows the scope of this application recommendation.

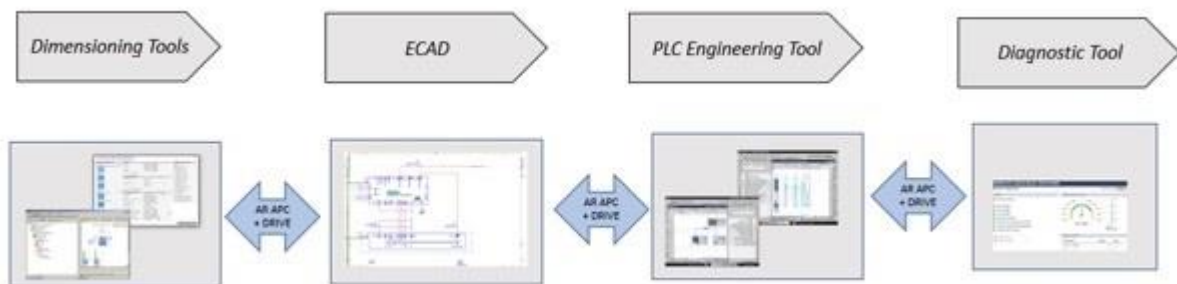


Figure 1 – Data exchange for dimensioning, electrical design, programming and diagnosis tools.

Figure 2 shows the different aspects of drive configurations. In this document the hardware parts and the electrical wiring aspects are covered. The mechanical and functional aspects are covered in the Application Recommendation Drive for MCAD (AR Drive MCAD).

¹ with exception of some length information of wiring cable defined by mechanical components and placement of the different electrical components

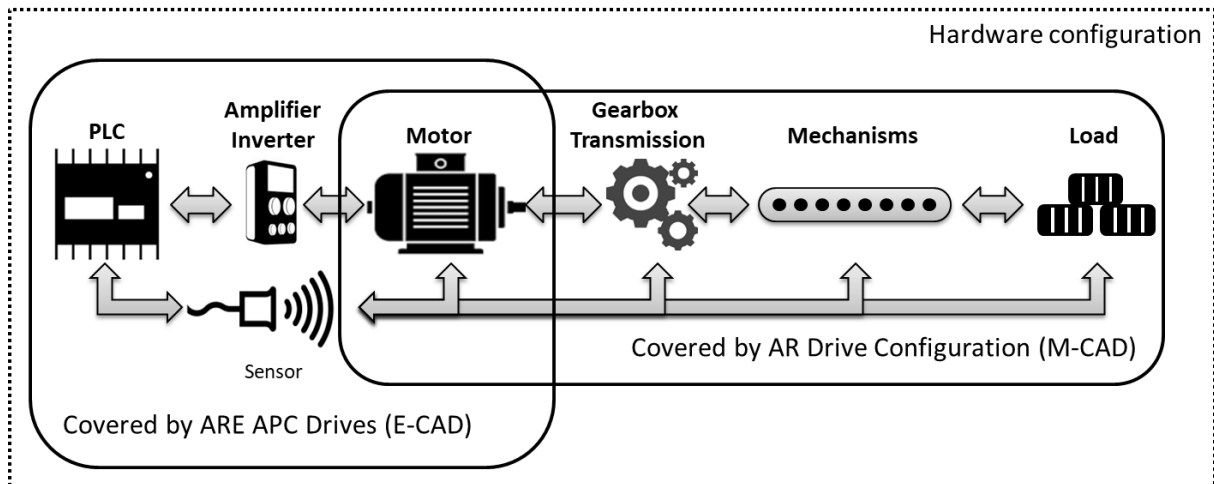


Figure 2 Aspects of drive configurations

Those different perspectives are covered in separate documents. The reasons for this separation are the following:

- Different users/engineering disciplines need different data and do not want to take care about the information that they don't need
- The data models that are described shall be usable standalone as well as interleaved
- It shall be possible to only create an update of one of them without effecting the other one

An additional chapter in the AR Drive MCAD Configuration document covers the topic of interleaving the data models of the different aspects.

2.1 Data exchange workflow

Usually the configuration of a system or machine using drives starts with a functional view of system which also defines the dynamic aspects. This design process can be supported by a machine concept design tool. Based on this information the mechanical and electrical configuration of the systems starts, where the corresponding electrical elements will be chosen to fulfil the functional and dynamic aspects of the system. This process may also be supported by a hardware selection tool.

2.2 Recommended workflow

According to the described criteria in most cases the following workflow is established.

- Set up a functional view of the drive system which defines its functional and dynamic aspects.
- Select electrical hardware modules, which meet the derived requirements.
- Set up the machine configuration and networking with a configuration tool.
- Import this configuration to ECAD tool, engineering of the ECAD project, and exporting the ECAD project to PLC programming tool
- Importing ECAD project into PLC programming tool and engineering of the overall PLC project

2.2.1 ECAD engineering

Based on the existing ECAD project, the ECAD engineer executes the complete hardware construction of the automation system including the drive configuration.

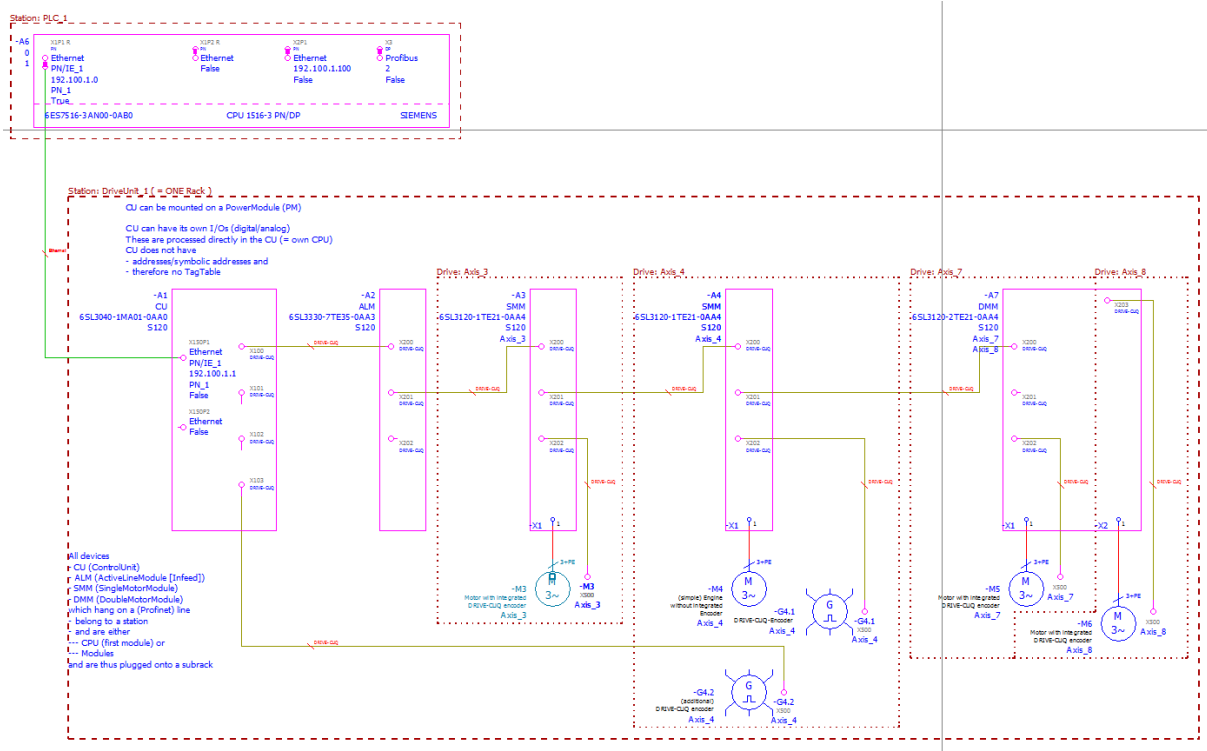


Figure 3 – ECAD engineering of a multi-axis drive system

2.2.2 PLC engineering

From ECAD tool the hardware configuration will be imported into the PLC engineering tool.

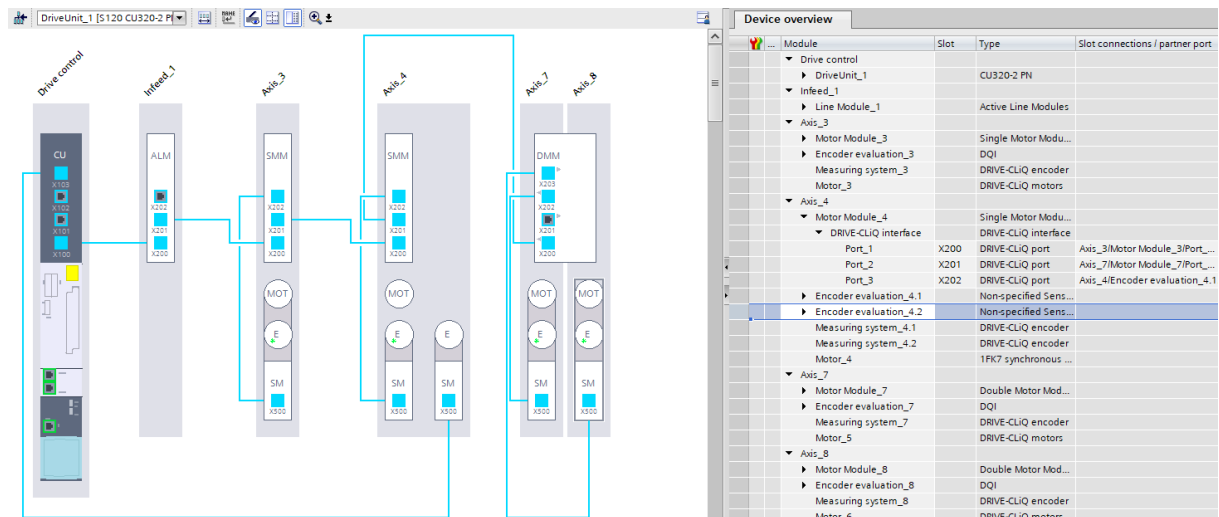


Figure 4 – Hardware configuration of a multi-axis drive system within engineering tool

On a next step the software program will be linked to the hardware by

1. Matching the symbolic addresses of the program to the symbolic names of the hardware

2. Assigning the axis software objects used for controlling the movement of an axis to the drive's objects which act as an interface to the hardware and firmware of the drive configuration.

Naming conventions:

To achieve an automatic assignment of the axis to the drive object, it is good practice to assign them the same name or names where the two elements can easily be recognized as being associated.

Name of the drive which controls a motor should include name of the motor to reflect the association between the drive and the motor.

2.2.3 Drive engineering

Adapting the drive configuration will be done by parametrizing all hardware modules and the drive configuration itself. This will be done in parallel to the PLC programming, and the final adaption will be done during commissioning of the system or machine.

3 Drive Configuration data structures in AutomationML

This application recommendation is based on the definitions provided by Automaton Project Configuration (see /AR APC V1.2.0.0/).

All basic concepts and definitions of the Application Recommendation “Automation Project Configuration” are also valid for this document, if not described differently.

Only added and changed data structures and objects will be described within this document.

3.1.1 Drive Configuration data exchange data model

The consideration of all these mentioned and already existing models leads to the following Automation Project Configuration data exchange diagram:

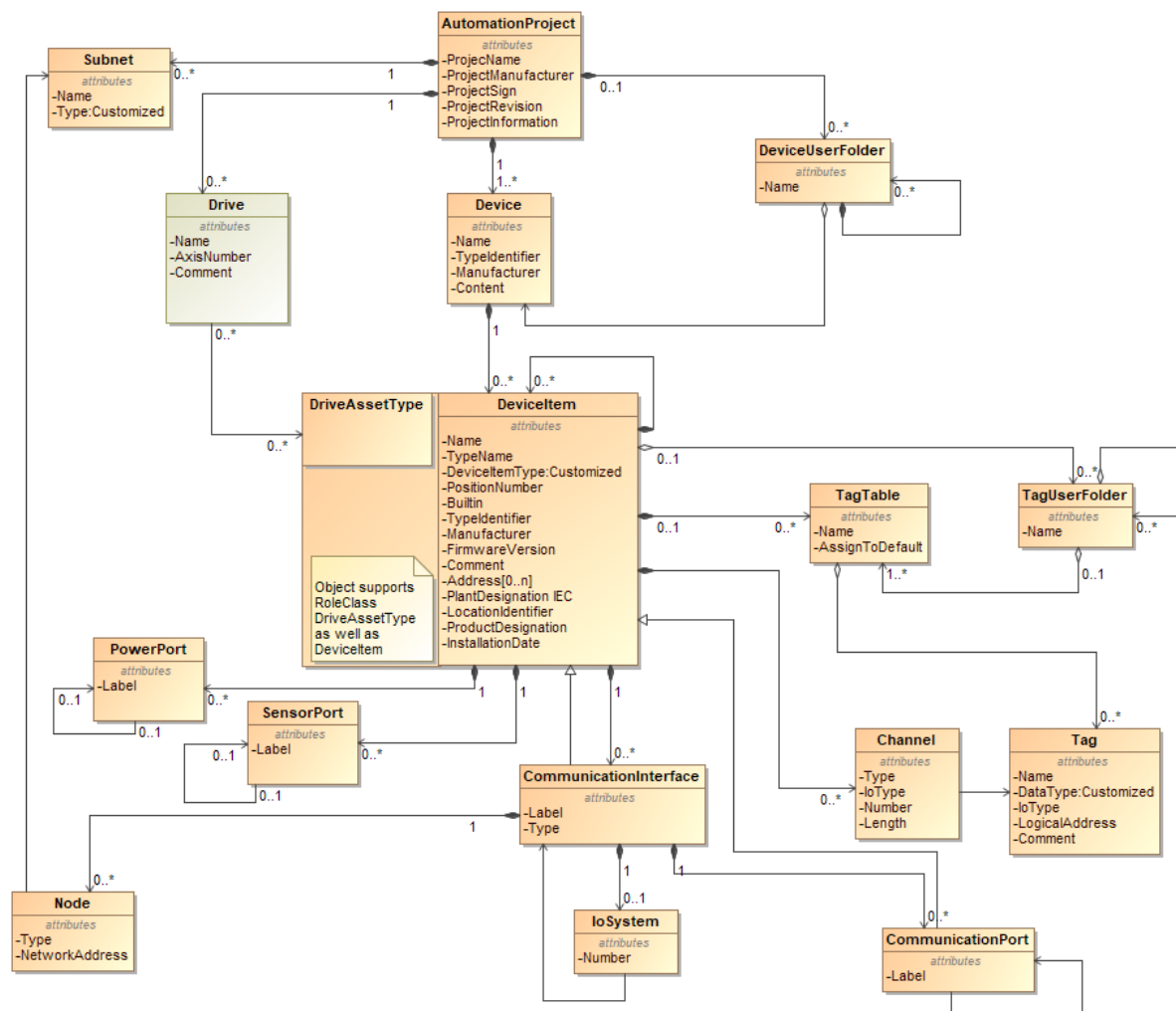


Figure 5 – Objects and parameters of the Drive Configuration data exchange and their relation to the elements of Automation Project Configuration

3.1.1.1 Drive

A Drive object holds references to the DriveAssetTypes, which represents the electrical components of the drive. The overall dynamic behaviour of the drive configuration may be controlled by a software object, which is identified by the name of the drive.

For a PLC program, this object may also act as an interface object to the drive functionality and cyclic I/O. The standard parameters of a Drive object are listed below.

- Name (string)
The name of the Drive shall be unique within the automation project.
- AxisNumber (int)
Optional information, which identifies the axis the drive object is related to..
- Comment (string):
An optional comment for the device, using a multilanguage string

The Drive objects are aggregated directly under the AutomationProject.

3.1.1.2 DriveAssetType

A DriveAssetType is an object to extend the definition of a DeviceItem with drive-specific information of the corresponding DeviceItem. It is also used to include the interface, which is used to reference the assigned Drive object.

In later Steps the „DriveAssetType“ could be used to specify the asset type more precisely, which is provided by the DeviceItem.

3.1.1.3 DriveAssignment

The interface class "Drive Assignment" is used to define the assignment of DriveAssetType to a Drive object. By using an internal link between the "Drive Assignment" interfaces of the Drive object and the DriveAssetTypes, the components are defined, which provide a dedicated functionality for the Drive.

4 Modelling of Drive Configuration data with AutomationML

4.1 RoleClassLibrary

Basement of the modelling are the required role classes. Facing the required model elements there are role classes especially required for Drive Configuration data modelling derived from role classes used for automation system modelling defined in Application Recommendation "Automation Project Configuration"

The following figures represent the defined role class library:

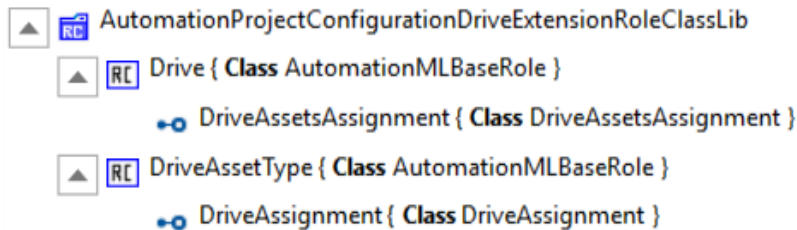


Figure 6 – AutomationProjectConfigurationDriveExtensionRoleClassLib in AutomationML Editor view

```

<RoleClassLib
  Name="AutomationProjectConfigurationDriveExtensionRoleClassLib"
  ChangeMode="state">
    <Version
      ChangeMode="state">1.2.0</Version>
    <RoleClass
      Name="Drive"
      RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole"
      ChangeMode="state">
        <Attribute
          Name="Name"
          AttributeDataType="xs:string"
          ChangeMode="state"/>
        <Attribute
          Name="Comment"
          AttributeDataType="xs:string"
          ChangeMode="state"/>
        <Attribute
          Name="AxisNumber"
          AttributeDataType="xs:int"
          ChangeMode="state"/>
        <ExternalInterface
          Name="DriveAssetsAssignment"
          RefBaseClassPath="AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssetsAssignment"
          ID="6832f132-d944-4abc-b2ce-a802e9aa8578"
          ChangeMode="state"/>
      </RoleClass>
    <RoleClass
      Name="DriveAssetType"
      RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole"
      ChangeMode="state">
        <ExternalInterface
          Name="DriveAssignment"
          RefBaseClassPath="AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssignment"
          ID="2f157da3-224d-471c-86ec-cc98cfb1d7d6"
          ChangeMode="state"/>
      </RoleClass>
    </RoleClassLib>

```

Figure 7 – AutomationProjectConfigurationDrivesExtensionRoleClassLib as XML representation

4.1.1 Drive

A “**Drive**” is derived from “AutomationMLBaseRole” according to AutomationML Whitepaper - Architecture and general requirements. It is defined as follows.

Table 2 – Definition Drive

Role class name	Drive	
Description	The role class “Drive” shall be used in order to support the structure of a Drive configuration within a project.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for Element reference	AutomationProjectConfigurationDriveExtensionRoleClassLib/Drive	
Attributes	“Name” (AttributeDataType=“xs:string”)	The attribute “Name” defines the name of the Drive. This attribute is mandatory. <i>Note:</i> This attribute is modelled by the standard attribute Name of the relevant CAEX object. <i>Note:</i> This attribute shall be unique within the automation project and aligned to the name of the axis within the PLC program. For drives referencing motors the name should be aligned to the name of the motor DeviceItem
	“Comment” (AttributeDataType=“xs:string”)	The attribute “Comment” defines a comment for the drive. The attribute “Comment” follows the Best Practice Recommendation Multilingual expressions in AutomationML. This attribute is optional.
	“AxisNumber” (AttributeDataType=“xs:int”)	The attribute “AxisNumber” defines the number of the axis the Drive is assigned to. This attribute is optional.
Interfaces	“DriveAssetsAssignment” (RefBaseClassPath=“AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssignment”)	This interface is used to link the Drive to the DriveAssetTypes, which represent the physical components of the drive. Exactly one DriveAssignment is allowed.

4.1.2 DriveAssetType

A “**DriveAssetType**” is derived from “AutomationMLBaseRole”. It is defined as follows.

Table 3 – Definition DriveAssetType

Role class name	DriveAssetType	
Description	The role class "DriveAssetType" shall be used in order to extend a DeviceItem to hold the interface for its assignment to a Drive object.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for Element reference	AutomationProjectConfigurationDriveExtensionRoleClassLib/DriveAssetType	
Attributes	None	No attributes in V 1.0.0
Interfaces	"DriveAssignment" (RefBaseClassPath="AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssignment")	This interface is used to link the Drive to the DeviceItem, which represent the physical components of the drive. Several DriveAssignments are possible. The direction of the link is not relevant. In case of connection to more than one DriveAssignments the one and only DriveAssignment shall contain all connections.

4.2 InterfaceClassLibrary

The next main base of modelling are interface classes. Facing the required model elements there are interface classes especially required for Drive data modelling derived from interface classes used from CommunicationRoleClassLib V 1.0.1 as defined in AutomationML Whitepaper – Communication.

The following figures represent the interface class library.

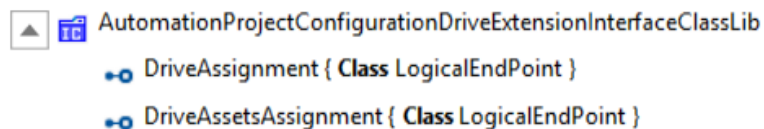


Figure 8 –AutomationProjectConfigurationDriveExtensionInterfaceClassLib in AutomationML Editor view

```

<InterfaceClassLib
  Name="AutomationProjectConfigurationDriveExtensionInterfaceClassLib"
  ChangeMode="state">
  <Version
    ChangeMode="state">1.2.0</Version>
  <InterfaceClass
    Name="DriveAssignment"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint"
    ChangeMode="state"/>
  <InterfaceClass
    Name="DriveAssetsAssignment"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint"
    ID="cfb4ea0b-dab8-4116-86b3-4f4fb0b1b148"
    ChangeMode="state"/>
  </InterfaceClassLib>

```

Figure 9 – AutomationProjectConfigurationDriveExtensionInterfaceClassLib as XML representation

Each Drive object has exactly one external interface of the class DriveAssetsAssignment which has internal links to all of the associated DriveAssetTypes.

A DriveAssetType has exactly one external interface of the class DriveAssignment which has internal links to the Drive object that it is associated with.

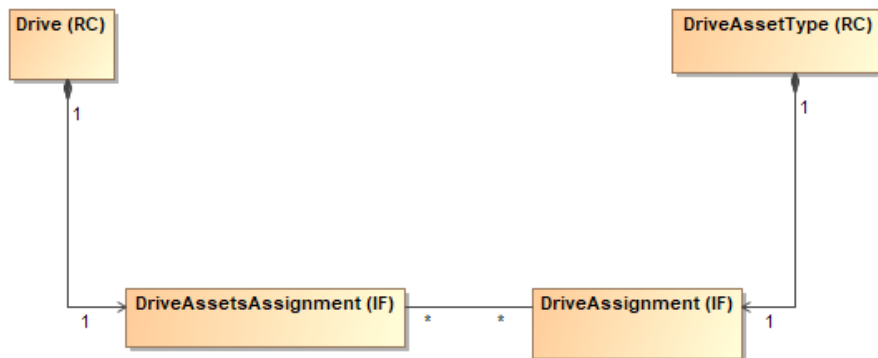


Figure 10 – Assigning Drive and DriveAssetType

4.2.1 DriveAssignment

A “**DriveAssignment**” is derived from a “LogicalEndpoint” according to AutomationML Whitepaper - Communication. It is defined as follows.

Table 4 – Definition DriveAssignment

Role class name	DriveAssignment
Description	The Interface class “DriveAssignment” shall be used in order to represent the logical assignment of a drive to its different DriveAssetTypes which represent the actual hardware components that form the drive train.
Parent Class	CommunicationInterfaceClassLib/LogicalEndPoint
Path for Element reference	AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssignment

The interface class “Drive Assignment” is used to define the assignment of a Drive to the different DriveAssets, which represent the electrical components of the drive.

4.2.2 DriveAssetsAssignment

A “**DriveAssetsAssignment**” is derived from a “LogicalEndpoint” according to AutomationML Whitepaper - Communication. It is defined as follows.

Table 5 – Definition DriveAssetsAssignment

Role class name	DriveAssetsAssignment
Description	The Interface class "DriveAssetsAssignment" shall be used in order to represent the logical assignment of a DriveAsset to its different mechanical DriveAssetTypes which represent the actual hardware components that form the drive train.
Parent Class	CommunicationInterfaceClassLib/LogicalEndPoint
Path for Element reference	AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssetsAssignment

The interface class "DriveAssetsAssignment" is used to define the assignment of a DriveAsset to the mechanical parts to be defined in AR Drives.

Appendix A Appendix A: Automation-ML examples

A.1 Drive Configuration with infeed, inverter and sensor modules (Example of PLC with S120)

This example shows a modular multi-axis drive with 4 drive objects consisting of control unit, infeed, inverters and motors with attached measurement systems.

The different Drive objects are linked to the device items which represent the electrical parts of the drive.

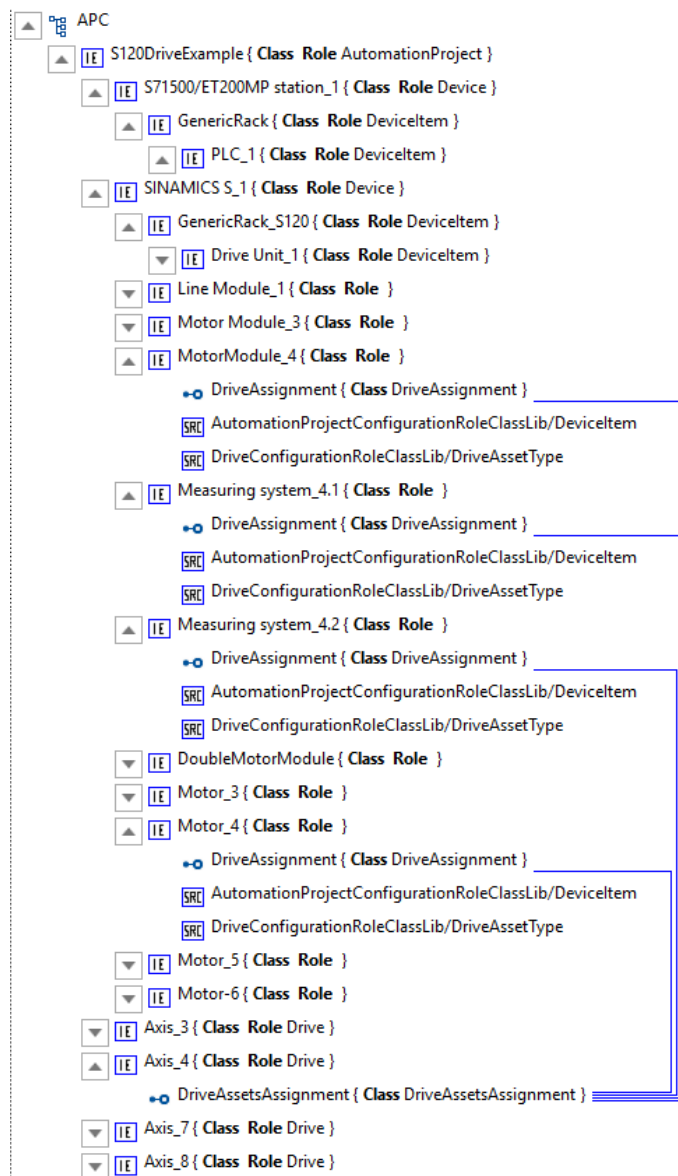


Figure 11 – Hardware configuration of a multi-axis drive system within engineering tool as XML representation

A.2 Drive Configuration with double multi-motor-module

This example shows a drive configuration including a multi-axis servo amplifier that controls two motors.

The multi-axis amplifier is linked to the corresponding Drive objects using the DriveAssignments interfaces and internal links.

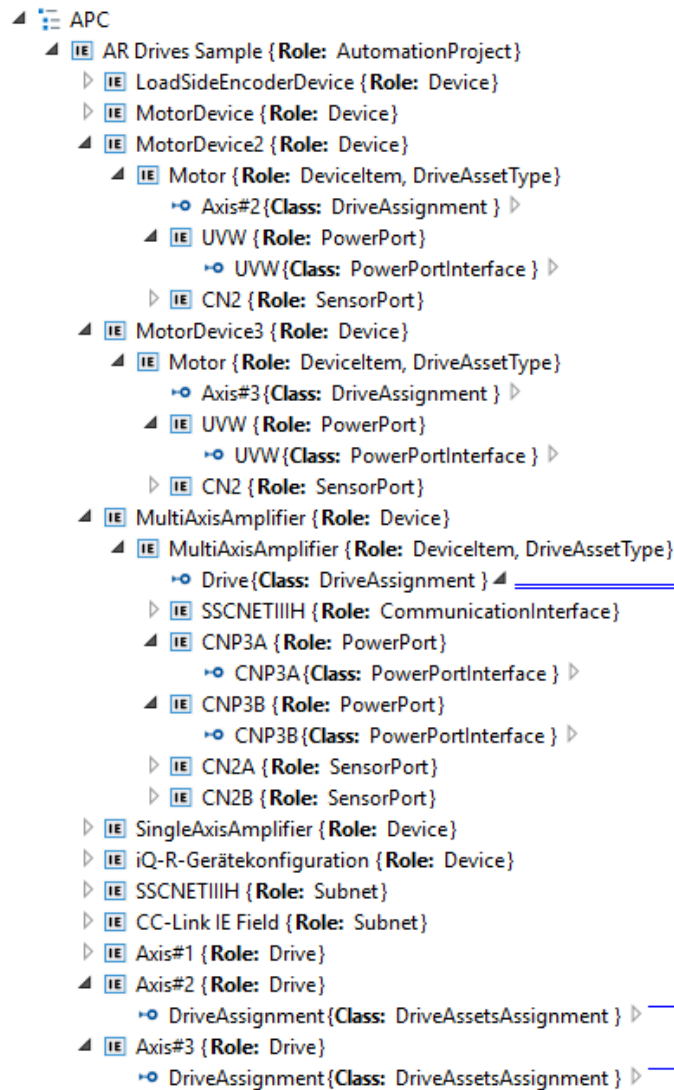


Figure 12 – Hardware configuration of a double-motor-module within engineering tool as XML representation