<AutomationML/>

The Glue for Seamless Automation Engineering

Conventions for modelling AutomationML libraries

State: November 2023

Table of contents

1 Conventions for the modelling of AutomationML libraries

1.1 Motivation and use cases

This document is intended for data modellers and their working groups who wish to create and publish AutomationML libraries. This includes libraries published by the AutomationML association, but also user-defined AutomationML libraries developed in industrial or academic environments.

Application-specific harmonised AutomationML class libraries are the basis for perfoming data exchange because they ensure that instances and data structures are based on a common understanding.

Since AutomationML, as a flexible object-oriented data description language, offers a wide range of possibilities for structuring class libraries, naming their elements and versioning them, the AutomationML association has compiled recommendations on how to create libraries in this document. The conventions described are based on the experience of AutomationML working groups and are intended to provide guidance for modellers inside and outside the AutomationML association when modelling AutomationML libraries.

1.2 Terms defining upper and lower case letters

The following section describes different ways of capitalising identifiers. These terms are referred to the remainder of the document. Excluded from these rules are units and proper names such as

- kg, kWh (SI unit symbols and their derivatives)
- IOSB, ARD, ZDF, ABB etc.

1.2.1 Pascal Casing

In this convention, the first character of each word is capitalised, as shown in the example below.

Examples:

- Correct: RollConveyor
- Wrong: Rollconveyor, Roll_Conveyor

1.2.2 Lower Camel Casing

In this convention, the first letter of each word is capitalised, except for the first word, as in the following example.

Examples:

- Correct: backColor
- Wrong: BackColor, backcolor

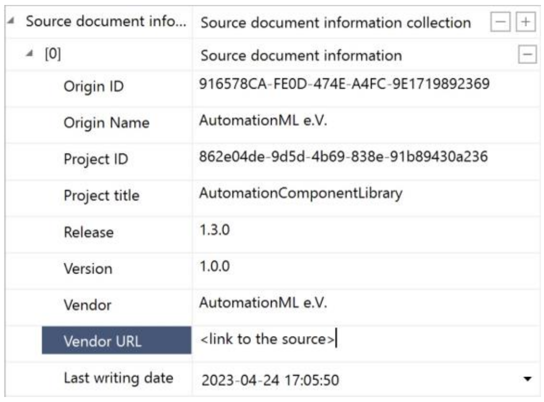
1.2.3 UpperCase

In this convention, all letters of a word are capitalised.

Examples:

- Correct: IP, NR, KITKARLSRUHE, ADAC
- Wrong: KITKarlsruhe

1.3 Recommendations for self-identification of libraries and extensions

ID	Requirement	Motivation	Priority
A_NM_01	**Mandatory:** An AutomationML library file must identify itself in the CAEX document. If there is only one library in the library file, the the following fields of the *CAEX SourceDocumentInformation* must be filled in and identify this libary. - OriginID: unique identifier for the library - OriginName: unique name for the library - OriginVersion: version number of the library - LastWritingDateTime: date of the release - OriginVendor: name of the producer or producer group - OriginVendorURL: URL to the library (collection) e.g.  Source document info... — Source document information collection [0] — Source document information Origin ID — 916578CA-FE0D-474E-A4FC-9E1719892369 Origin Name — AutomationML e.V. Project ID — 862e04de-9d5d-4b69-838e-91b89430a236 Project title — AutomationComponentLibrary Release — 1.3.0 Version — 1.0.0 Vendor — AutomationML e.V. Vendor URL — <link to the source> Last writing date — 2023-04-24 17:05:50 • The URL points to an online repository where the original library can be found, including documentation if applicable. The link is complete and points to downloadable content.	The aim is to assign the library to a producer group and to define the version of the release. The URL points to a download source of the original library, if available.	**Mandatory**
A_NM_02	In addition to A_NM_01: If serveral related libraries (one or more library collections) are stored together in an AutomationML document, the *SourceDocumentInformation* field is to be filled in for each collection of the contained libraries, i.e. one entry per related libraries. • Each collection receives its own *OriginVersion*, which is determined by the working group.	Improving the traceability and downloadability of the libraries.	**Mandatory**
A_NM_03	**Mandatory**: AutomationML standard libraries are signed. **Recommendation**: User-defined libraries are signed. That means that, conversely, every change to a library leads to the signature being broken. It follows from this that once a library has been signed and published, its version can no longer be changed without being noticed.	The signature protects a library from unwanted modification or allows information about new versions to be rolled out.	**Mandatory** for AutomationML standard libraries **Recommendation** for user-defined libraries

1.4 Recommendations for library naming conventions

1.4.1 Naming conventions for libraries

ID	Requirement	Motivation	Priority
A_NKL_01	**Mandatory: General rules for library names** • Names of libraries must be unique within an AutomationML document. • A library name is composed of the following components in the given order: Prefix | Domain | Type • Name components are to be separated with a separator "_".The underscore separator "_" must NOT be used for any other purposes. • The naming of each component is done with PascalCasing. • Prefix, domain and type may only contain the letters [a-z], [A-Z], [-] as well as [0-9], no spaces. Examples of library names: • AutomationML_ComponentBase_InterfaceClassLib • AutomationML_AutomationProjectConfiguration_RoleClassLib • HSPF_MasterClass_AttributeTypeLib • ABB_RobotSystem_SystemUnitClassLib	Quick identification of relevant information	**Mandatory**
A_NKL_02	**Mandatory**: Design of the prefix for AutomationML standard libraries **AutomationML standard libraries** • Start with the prefix "AutomationML", Example: AutomationML_PumpenLib…	Common Look & Feel, recognitionning effect	**Mandatory**
A_NKL_03	**Mandatory**: Design of the **prefix** for libraries that are not AutomationML standard libraries: user defined **AutomationML-libraries** • Start with a prefix that identifies the source of the library, e.g. an association name, company name, etc. The name is determined by the producer group. • The prefix "AutomationML" is reserved for standard libraries under the responsibility of the AutomationML Association. All other standard libraries must NOT start with the prefix "AutomationML" • Examples: Prefix: "IOSB", "Festo", "HSPF", etc.	Simple assignability of the libraries	**Mandatory**
A_NKL_04	**Mandatory: Design of the "domain"** • The domain name should identify the application for which the library was developed. • The naming is done by the producer group. e.g. Domain = "AutomationProducts", "AutomationComponent"	Improved readability	**Mandatory**
A_NKL_05	**Mandatory: Design of "type"** • Role libraries receive the identifier "RoleClassLib". • Interface libraries receive the identifier "InterfaceClassLib". • Attribut type libraries receive the identifier "AttributTypeLib". • SystemUnitClass libraries receive the identifier "SystemUnitClassLib".	Quick identification of the library type	**Mandatory**
A_NKL_06	**Mandatory:** Naming conventions for new versions of libraries • When a new version of a library is created, it keeps the name of the previous version.	Libraries must have a unique name. Uniqueness for references	**Mandatory**

1.4.2 Naming conventions for AutomationML files for libraries

ID	Requirement	Motivation	Priority
A_NKD_01	**Mandatory**: File names of AutomationML libraries must have an unique name that is unchangeable during the life cycle of the library. • A file name for an AutomationML library is built up from name elements in a specified order. • A separator "_" must be used between the name components. The underscore separator "_" must NOT be used for any other purpose. If only ***one*** AutomationML library is included in the AutomationML document: • The following name elements must be used | LibName | AMLEdition | LibVersion | .aml | • The *LibName* corresponds to library names. • The *AMLEdition* identifies the underlying AutomationML standard. ○ AutomationML Edition 1: "AMLEd1" ○ AutomationML Edition 2: "AMLEd2" • The *LibVersion* corresponds to the version of the library, format a.b.c. • The file extension is ".aml". • **Example:** AutomationML_ComponentBase_InterfaceClassLib_AMLEd2_1.1.0.aml If ***several*** libraries are contained in one AutomationML file: • The following name componets must be used | Prefix | Domain | „Libraries" | AMLEdition | Release-Identifier | .aml | • For the *Prefix* and the *Domain*, the same rules apply as described in 1.4.1. • This is followed by the string "*Libraries*", • The *AMLEdition* identifies the underlying AutomationML standard ○ AutomationML Edition 1: "AMLEd1" ○ AutomationML Edition 2: "AMLEd2" • The *Release-Identifier* identifies the version of the library collection, it is identical to the *CAEX OriginVersion* in the associated *SourceDocumentInformation*. This release indentifier is defined by the working group developing the library. • Examples: ○ AutomationML_ARAPC_Libraries_AMLEd1_1.3.0.aml ○ AutomationML_Part6WPCompo_Libraries_AMLEd1_1.1.0.aml	This makes it possible to distinguish between different versions of libraries just by looking at the file name. Otherwise, the libraries could simply be exchanged unnoticed.	**Mandatory**
A_NKD_02	**Mandatory**: File names of AutomationML standard library files • The naming of each component is done with PascalCasing. • The individual parts of the name may only contain the letters [a-z], [A-Z], [-] and [0-9], no spaces. **Optional**: For user-definded libraries, this is a recommendation.	Improved readability	**Mandatory** for AutomationML standard libraries **Recommendation** for user-defined libraries

1.4.3 Naming conventions for classes

ID	Requirement	Motivation	Priority
A_NKK_01	**Mandatory:** Names of classes (not Attribute Types) • Class names are named via PascalCasing.	Ensure verifiability	**Mandatory**
A_NKK_02	**Recommendation:** Classes should not have their type in their name. The type is known by the software and the libraries also indicate the type. • Correct: Roboter • Wrong: Roboter_SUC • Wrong: Roboter_SystemUnitClass	Advantage: the type is clearly recognisable.	**Recommendation**
A_NKK_03	**Recommendation**: If a new version of a class is created, it must be modelled in a new library (otherwise the signatory of the original library would become invalid). The name of the new version of the class should remain the same • **Example**: AutomationML_Pump_RoleClassLib (in version 1.0) • Pump_Superfast AutomationML_Pump_RoleClassLib (in version 1.1) • Pump_Superfast **Note**: The renaming of a new version of a class is nvertheless permitted, e.g. AutoFaceLift2010, AutoFaceLift2024	Causes better clarity in the referencing of classes. Attention: not automatically testable	**Recommendation**

1.4.4 Naming conventions for attributes and attribute types

ID	Requirement	Motivation	Priority
A_NKA_1	**Attribute names** **Recommendation**: Names of attributes or attribute types should be written using CamelCase. • referenceTpe, speedValue	Common coding guidelines	**Recommendation**
A_NKA_2	**Attribute data type names** • **Mandatory**: all attribute data types must be named as defined in: http://www.w3.org/TR/xmlschema-2/#built-in-datatypes • **E.g. xs:double**	Compatibility and dissemination	**Mandatory**
A_NKA_3	**Naming of units** **Recommendation:** Since a XML document cannot contain superscripts as in m^2, the substitution m^2 should be used instead. Since we are writing to an international audience, the values and units of attributes should be expressed in the metric system (SI). As a reference for the definition of SI units and symbols, the https://www.nist.gov/pml/owm/metric-si/si-units should be used. **Mandatory**: the characters */+-^ are reserved for arithmetic operations.	Tool and language independent readability	**Recommendation**
A_NKA_4	**Notation of composite units** **Mandatory:** the separation between numerator and denominator can be introduced by "/". For ambiguos expressions use parentheses, e.g. kg*m/s^2, kg*m/s^(-2), J/(A*s)	Tool independent readability	**Mandatory**

1.5 Recommendations for versioning classes and libraries

ID	Requirement	Motivation	Priority
A_V_01	**Mandatory**: Libraries must be versioned. For this purpose, the CAEX attribute *Version* must be set.	Preparation of sustainable development.	**Mandatory**
A_V_02	**Recommendation**: Classes must be versioned. For this purpose, the CAEX attribute *Version* must be set.	Preparation of sustainable development.	**Recommendation**
A_V_03	**Mandatory**: The version number must follow the concept of semantic versioning (see https://semver.org). a=Major Release, breaks compatibility b=Minor = Supplement, backwards compatible c=Patch = Working version backwards compatible • Libraries that are not downward compatible increment the major release a. • Downward compatible libraries increment b. • Working versions increment c. For pre-release versions, any pre-release tags can be added to the version number ”-alpha0.1“.	Faciliates the guided software-supported tracing of the version history.	**Mandatory**
A_V_04	**Mandatory**: If a library, type or class is modelled in a new version, the old and new version should reference each other. For this purpose the old version refers to the new version in the filed Revision/newVersion, wheras the new version refers to the old version in the field Revision/oldVersion. If a new version breaks with the previous one, i.e. goes beyond a change in the content of the major version, it is modelled as a new independent class and does not have to point to the previous version. If a library is converted from an old AutomationML edition to an AutomationML edition, the old and new libraries should alsi refer to each other.	Faciliates the guided software-supported tracing of the version history as well as the distribution, recognition, updating or informing about new versions.	**Mandatory**
A_V_05	**Mandatory**: If an AutomationML standard library is modified within an AutomationML edition, it must be published in a new version. An extension of existing libraries may never be made without affecting their version.	Easy recognition of changes in the library.	**Mandatory**
A_V_06	**Mandatory**: If an AutomationML library is converted from an old to a new AutomationML standard (e.g. from AutomationML Ed.1 to Ed.2), the version number of the new library is independent of the version number of the original library and can, for example, start from the beginning. For reasons of better comparability, the version number of libraries of different editions can be chosen to be identical if the contents are the same. The versions of both libraries can subsequently develop independently, so it cannot be concluded from the equality of the library version number across AutomationML edition boundaries that they are identical in terms of content. Note: The AutomationML edition is not recognised via the version number, but via other AutomationML mechanisms.	Distinctivness of the libraries, even if they do not differ in content.	**Mandatory**
A_V_07	**Mandatory**: The new library version must be saved in a new document and must not be in the same document as the older version.	Uniform separation of versions.	**Mandatory**

1.6 Recommendations for structuring libraries

ID	Requirement	Motivation	Priority
A_S_01	**Recommendation**: In official AutomationML libraries, tree structures should only be used if they correspond to the derivation hierarchy of the classes. **Recommendation**: It is recommended to model classes preferably in libraries as a flat list structure and not artificially reproduce the inheritance hierarchy. Instead, the inheritance hierarchy is to be created by the AutomationML Editor as a "view" can be generated.	The class tree is only one view of the library. To reduce confusion between the use of library hierarchy and class hierarchy. Problem: users often think that the tree structure corresponds to the class hierachy. This will improve understanding.	**Recommendation**
A_S_02	**Recommendation**: When a child class is created in a class, the child class should by default receive an inheritance relationship to the parent class by referencing the parent class.	Simplifies understanding by the user.	**Recommendation**
A_S_03	**Mandatory**: All libraries published by AutomationML working groups must contain only their own libraries. Referenced third-party libraries must be included via CAEX ExternalReferences and stored in separate files. **Recommendation**: It is also recommended for user-defined AutomationML libraries to be saved in separate files and to integrate those libraries in AutomationML projects by referencing.	This prevents modifications being made in copies of these standard libraries. It also reduces the size of the files.	**Mandatory** for AutomationML standard libraries **Recommendation** for user-defined libraries

1.7 Recommendations for the extension of libraries (extensions)

ID	Requirement	Motivation	Priority
A_E_01	When a working group extends a standard library, one of two ways described below must be chosen. Variant 1: If a standard library is modified, the working group publishes a new version of the library, e.g.: • AutomationML_Subsea_RoleClassLib (in version 1.0.0) • AutomationML_Subsea_RoleClassLib (in version 1.1.0) Variant 2: If a few classes are to be added to a standard library and the original library can still be used on its own, theses additional classes can be modelled in a separate library with its own versioning. • HSPF_ConferenceMonitor_InterfaceClassLib • HSPF_ConferenceMonitor_InterfaceClass_BluetoothExtension In variant 2 the extension is to be added to the original library name. The naming should reflect the application and the source library, for example with the text „BluetoothExtension“, see A_E_02. Software pattern: "Plugin-Pattern"	The advantage of an extension is that the basic library does not have to be published / documented / signed etc. when it is added.	**Mandatory**
A_E_02	For variant 2, the following rules apply to the naming of the extended library: LibName ExtPrefix „Extension" • *LibName* corresponds to the name of the original library • *ExtPrefix* is to identify the owner or owner group and the type of extension: This can be omitted if an owner and domain are identical to the original library. • *"Extension"* identifies that it is an extension of an existing library.	Uniform naming of the extended libraries and intuitive assignability and understanding of where these were derived.	**Mandatory for variant 2**

	e.g. • AutomationML_GoldFisch_InterfaceClassLib • AutomationML_GoldFisch_InterfaceClassLib_SilverExtension • AutomationML_GoldFisch_InterfaceClassLib_IOSBBronceExtension		
A_E_03	Extensions of extensions are not allowed. If an extension is not sufficient, a new independent extension must be created.	Better coordination of extensions, avoidance of wild growth.	**Mandatory**
A_E_04	The extension library should name both its own source and the original extended library in the *SourceDocumentInformation* field. Thus, it is known which library version is being extended.	Machine-readable traceability of which library is being extended.	**Mandatory**

1.8 Recommendations for the online-supported search of libraries, classes and types

ID	Requirement	Motivation	Priority
A_OM_1	**Mandatory:** AutomationML libraries, if they have gone through a publication process, are available online, findable online and accessible online by the AutomationML Editor. This requires online access. Offline operation is possible after downloading the libraries.	Online access is a prerequisite for all subsequent features.	**Mandatory**
A_OM_2	**Recommendation:** AutomationML attribute types, attributes, role libraries and role classes should become globally referenceable (analogous to IRDI). AutomationML libraries should themselves be able to become a referencable semantic standard. • SemanticRef = amlwebsite	Allows AutomationML semantics to be referenceable.	**Recommendation**
A_OM_3	**Recommendation**: If a new attribute type is defined and it is found that the attribute type already exists, the new attribute type should not be used. The existing attribute type should be used instead, to prevent duplication of attribute types.	A user-defined library can be built from several sources without duplication of types.	**Recommendation**
A_OM_4	**Recommendation**: If a new class is defined and it already exists and is available online, it should not be defined again but reused.	Avoidance of duplication of classes. Reuse of classes.	**Recommendation**
A_OM_5	**Recommendation**: An AutomationML document with standard libraries should internally store a path to its original digital source. E.g. AMLStandardLib store a URI to www.automationml.org/... (the place where this library can be found)	The path is set by the AutomationML Office.	**Recommendation**

1.9 Modelling recommendations of libraries, classes and attribute types

ID	Requirement	Motivation	Priority
A_AE_ME_01	**Recommendation**: All attribute types should have a direct or indirect sematic reference to existing dictionaries (CC dictionary, semantic standards, e.g. other AutomationML classes), **if available.**	Automated interpretability of features.	**Recommendation**
A_AE_ME_02	**Mandatory**: CAEX 3.0: All attributes used in role classes and interface classes must be typed. All attributes must be derived from an AutomationML attribute type.	Prevents the use of untyped attributes.	**Mandatory**
A_AE_ME_03	**Recommendation**: Attribute types should always include: • Description (Language: English for AutomationML working groups) • Data type • Unit as defaults (see also A_AE_ME_04)	Desciption: important for human readability. Type safety: Whoever wants to change these values must overwrite the attribute.	**Recommendation**

A_AE_ME_04	**Mandatory**: • If the data type of an attribute is "xs:string" or the attribute is a structure attribute such as a list attribute, the unit must remain empty and is not evaluated. • If the data type is a numerical value, e.g. xs:int, xs:float, etc., the unit must always be specified, in case of non-existence with "ilb" (intentionally left blank).	Makes troubleshooting automatable.	**Mandatory**
A_AE_ME_05	**Mandatory** for AutomationML standard libraries, **recommended** for user-defined libraries: AutomationML libraries must have the following fields: • Description (language: English for AutomationML working groups) • Version (see A_V_01)	Transparency	**Mandatory** for AutomationML standard libraries **Recommendation** for user-defined libraries
A_AE_ME_06	**Recommendation**: Only AutomationML standard libraries that are signed should be referenced. Attention: during the development of a library this should be tolerated, but in final libarries it should not be tolerated.	Unsigned libraries are potentially manipulated. Note: Beta versions can also be signed.	**Recommendation**

1.10 Recommendations for rights managment of libraries

ID	Requirement	Motivation	Priority
A_R_1	Master libraries may only be changed by authorised persons. The rights system for this is to be defined.	Tamper protection IP Protetion Signing	**Mandatory**
A_R_2	Libraries must be proteced from unwanted modification.	Tamper protection	**Mandatory**

1.11 Recommendations for saving libraries and AutomationML documents

ID	Requirement	Motivation	Priority
A_S_1	**Mandatory**: If AutomationML documents are stored on a hard disk, third-party libraries must be stored in a subfolder whose name is "_AMLExternalLibraries“. This subfolder contains offline copies of the libraries used dierectly in the document folder. These libraries are to be referenced in the main document via a suitable alias. The referencing is done via a relative path. **Note**: Sending of all related documents can be done by means of container format AMLX. Missing libraries can be downloaded independently by the AutomationML Editor.	All foreign libraries are located in the subfolder, "_AMLExternalLibraries". This provides tamper protection, the avoidance of multiple loading of conflicting libraries, and creates smaller files.	**Mandatory**
A_S_2	**Mandatory**: All aliases must be named for referencing external libraries. The following rules apply: • An alias must be unique within the document. • **Implementation recommendation:** This can be achieved by having the alias correspond to the name of the target library (not the file). • **Implementation recommendation**: If a more compact form is desired, this is permitted as long as the uniqueness of the alias is maintained.	Human readability	**Mandatory**
A_S_3	**Mandatory**: The path must point to the libraries relative to the document folder.	Traceability	**Mandatory**

The implementaion is illustrated in Figure 1.

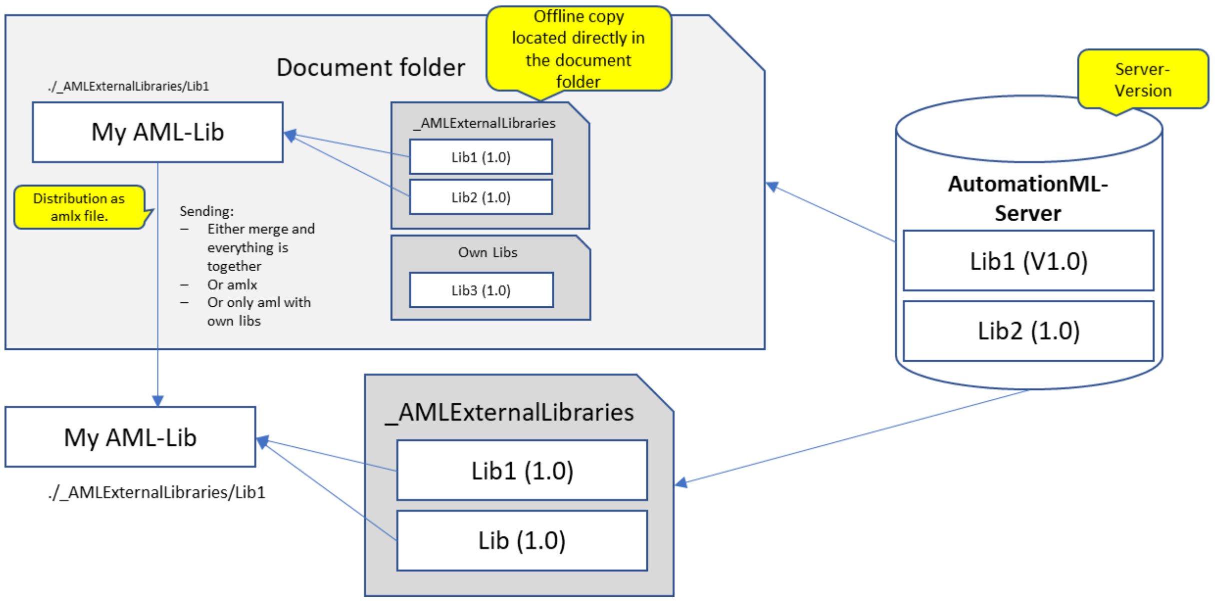


Figure 1: Distributed document architecture of an AutomationML document, e.g. an AutomationML library