<AutomationML/>

The Glue for Seamless Automation Engineering

Konventionen für die AutomationML Bibliotheksentwicklung

State: Februar 2025

Version 1.4.6

Inhaltsverzeichnis

1 KONVENTIONEN FÜR DIE MODELLIERUNG VON AML BIBLIOTHEKEN 3

1.1 Motivation und Anwendungsfälle 3

1.2 Begriffe zur Definition von Groß- und Kleinschreibung 4

1.2.1 Pascal Casing 4

1.2.2 Lower Camel Casing 4

1.2.3 UpperCase 4

1.3 Namenskonventionen für AutomationML-Bibliotheken 5

1.3.1 Namenskonventionen für Bibliotheken 5

1.3.2 Namenskonventionen für AML-Bibliotheks-Dateien 6

1.3.3 Namenskonventionen für Klassen 7

1.3.4 Namenskonventionen für Attribute und Attributtypen 7

1.4 Konventionen zur Strukturierung von Bibliotheken 8

1.5 Konventionen für die Modellierung von Bibliotheken, Klassen und Attributtypen 9

1.6 Konventionen für Erweiterungsbibliotheken (Extensions) 10

1.6.1 Namenskonventionen für Erweiterungsbibliotheken 10

1.6.2 Namenskonventionen für Erweiterungsbibliotheksdateien 11

1.6.3 Konventionen für die Modellierung von Erweiterungsbibliotheken 12

1.7 Konventionen zur Versionierung von Klassen und Bibliotheken 13

1.7.1 Begriffsbestimmung: Versionierung versus Vererbung 13

1.7.2 Begriffe VersionReference und ExtensionReference 13

1.7.3 Allgemeine Konventionen für das Modellieren von Versionen 14

1.7.4 Allgemeine Konventionen zur gegenseitigen Referenzierung von Versionen 16

1.7.5 Spezielle Konventionen zur Modellierung von Versionsreferenzen 16

2 KONVENTIONEN FÜR DAS PUBLIZIEREN UND DIE VERWENDUNG PUBLIZIERTER AML BIBLIOTHEKEN, KLASSEN UND TYPEN 18

2.1 Selbstidentifikation und Signierung von Bibliotheken und Extensions 18

2.2 Empfehlungen zur online-Verwendung von Bibliotheken, Klassen und Typen 19

2.3 Konventionen zum Speichern von referenzierten Bibliotheken und AML-Dokumenten 20

2.4 Rechteverwaltung von Bibliotheken 21

1 Konventionen für die Modellierung von AML Bibliotheken

1.1 Motivation und Anwendungsfälle

Dieses Dokument richtet sich an Datenmodellierer und ihre Arbeitsgruppen, die AutomationML Bibliotheken erstellen und veröffentlichen möchten. Dies umfasst sowohl Bibliotheken, die vom AutomationML-Verein herausgegeben werden, aber auch nutzerdefinierte AutomationML Bibliotheken im industriellen oder akademischen Umfeld entwickelt werden.

Anwendungsspezifische harmonisierte AutomationML Klassenbibliotheken sind Grundlage für die Durchführung von Datenaustausch, weil sie sicherstellen, dass die Instanzen und Datenstrukturen auf einem gemeinsamen Verständnis basieren.

Da AutomationML als flexible objektorientierte Datenbeschreibungssprache vielfältige Möglichkeiten bietet, um Klassenbibliotheken zu strukturieren, ihre Elemente zu benennen und zu versionieren, trägt der AutomationML Verein mit diesem Dokument Handlungsempfehlungen zusammen, wie man Bibliotheken erstellen soll. Die beschriebenen Konventionen basieren auf Erfahrungen von AML Arbeitsgruppen und sollen Modellierern innerhalb und außerhalb des AutomationML-Vereins Orientierung bei der Modellierung von AML Bibliotheken geben.

1.2 Begriffe zur Definition von Groß- und Kleinschreibung

Im folgenden Abschnitt werden verschiedene Arten der Großschreibung von Bezeichnern beschrieben. Auf diese Begriffe wird im weiteren Verlauf des Dokuments Bezug genommen. Ausgenommen von diesen Regeln sind Einheiten und Eigennamen wie

- kg, kWh (SI Einheitensymbole und ihre Ableitungen)
- IOSB, ARD, ZDF, ABB etc.

1.2.1 Pascal Casing

Bei dieser Konvention wird das erste Zeichen eines jeden Wortes großgeschrieben, wie im folgenden Beispiel dargestellt.

Beispiele:

- Richtig**: R**oll**C**onveyor
- Falsch: Rollconveyor, Roll_Conveyor

1.2.2 Lower Camel Casing

Bei dieser Konvention wird der erste Buchstabe jedes Wortes großgeschrieben, mit Ausnahme des ersten Wortes, wie im folgenden Beispiel.

Beispiele:

- Richtig: back**C**olor
- Falsch: BackColor, backcolor

1.2.3 UpperCase

Bei dieser Konvention werden alle Buchstaben eines Wortes großgeschrieben.

Beispiel:

- Richtig: IP, NR, KITKARLSRUHE, ADAC
- Falsch: KITKarlsruhe

1.3 Namenskonventionen für AutomationML-Bibliotheken

1.3.1 Namenskonventionen für Bibliotheken

ID	Anforderung	Motivation	Priorität
A_NKL_01	**Pflicht: generelle Regeln für die Benennung von AML-Bibliotheken** (Hinweis: für Erweiterungsbibliotheken gelten die in Abschnitt 1.6 beschriebenen Konventionen) • Namen von Bibliotheken müssen innerhalb eines AML Dokumentes eindeutig sein. • Ein Bibliotheksname wird aus folgenden Bestandteilen in der vorgegebenen Reihenfolge zusammengesetzt, sie werden im Folgenden näher erläutert: Prefix _ Domain _ Type • Namensbestandteile sind mit einem Separator „_“ zu trennen. Dieser darf nicht in anderen Namensbestandteilen vorkommen. • Die Ausgestaltung von Prefix, Domain und Type wird in den nachfolgenden Konventionen definiert. • Die Namensgebung jedes Bestandteiles erfolgt mit PascalCasing. • *Prefix*, *Domain* und *Type* dürfen nur die Buchstaben [a-z], [A-Z], [-], sowie [0-9] enthalten, keine Leerzeichen. Beispiele für Bibliotheksnamen: • AutomationML_ComponentBase_InterfaceClassLib • AutomationML_AutomationProjectConfiguration_RoleClassLib • HSPF_MasterClass_AttributeTypeLib • HSPF_MasterClass.2025_AttributeTypeLib • ABB_RobotSystem_SystemUnitClassLib • ABB_RobotSystem.Welding_SystemUnitClassLib	Leichte Identifizierbarkeit relevanter Informationen	**Pflicht**
A_NKL_02	**Pflicht**: Ausgestaltung des **Prefix** für AML Standardbibliotheken **AML Standardbibliotheken** • beginnen mit dem Prefix „AutomationML“, Beispiel: AutomationML_PumpenLib	Common Look & Feel, Wiedererkennungseffekt	**Pflicht**
A_NKL_03	**Pflicht**: Ausgestaltung des **Prefix** für Bibliotheken, die keine AML Standardbibliotheken sind: Nutzerdefinierte **AML-Bibliotheken** • beginnen mit einem Prefix, der die Quelle der Bibliothek identifiziert, z.B. ein Verbandsname, Firmenname usw. Der Name wird von der Erzeugergruppe festgelegt. • Alle NICHT vom AML-Verein verantworteten Standard-Bibliotheken dürfen NICHT mit dem Prefix „AutomationML“ beginnen. Beispiele: Prefix: „IOSB“, „Festo“, „HSPF“, etc.	Einfache Zuordenbarkeit der Bibliotheken	**Pflicht**

A_NKL_04	**Pflicht: Ausgestaltung der „Domain“**
	• Die Domain identifiziert bzw. klassifiziert die Anwendung, für die das AML-Dokument und alle seine enthaltenen Bibliotheken entwickelt wurde. Beispiele:
	  ○ „AutomationProducts“,
	  ○ „AutomationComponent“,
	  ○ “RobotSystems”
	• Die Namensgebung erfolgt durch die Erzeuger der Bibliothek.
	• Die Domain ist ein String, er kann bei Bedarf in mehrere durch einen Punkt „.“ separierte Bestandteile unterteilt werden. Sub-Domains bestimmen das Anwendungsgebiet näher.
	  ○ Die Syntax der Unterteilung ist Maindomain+“.“+Subdomain.
	  ○ Mehrere Subdomains sind erlaubt.
	  ○ Beispiele für unterteilte Domains sind:
	  ▪ „RobotSystem.Welding“,
	  ▪ „MasterClass.2025“,
	  ▪ „Robotics.TwoArms.Gripper“
	• In der Namensgebung der Domain ist die Angabe der Maindomain verpflichtend, die Angabe der Subdomains optional.
		Verbesserte Lesbarkeit, Klassifizierung und erleichtere Erkennung von Bibliotheken	**Pflicht**

A_NKL_05	**Pflicht: Ausgestaltung von „Type“** • Rollenbibliotheken erhalten den Bezeichner „RoleClassLib“. • Interfacebibliotheken erhalten den Bezeichner „InterfaceClassLib“. • Attributtypbibliotheken erhalten den Bezeichner „AttributeTypeLib“. • SystemUnitClass-Bibliotheken erhalten den Bezeichner „SystemUnitClassLib“.	Leichte Identifizierbarkeit der Bibliotheksart.	**Pflicht**

1.3.2 Namenskonventionen für AML-Bibliotheks-Dateien

ID	Anforderung	Motivation	Priorität
A_NKD_01	**Pflicht**: **generelle Regeln für die Benennung von AML-Bibliotheks-Dokumenten** (Hinweis: für Erweiterungsbibliotheken gelten die in Abschnitt 1.6 beschriebenen Konventionen) • Dateinamen von AML-Bibliotheken müssen einen eindeutigen und im Lebenszyklus der Bibliothek unveränderlichen Namen haben. • Ein Dateiname für eine AML-Bibliothek wird aus folgenden Bestandteilen in der vorgegebenen Reihenfolge zusammengesetzt. • Zwischen den Namensbestandteilen ist ein Separator „_“ zu verwenden. Dieser darf nicht in den übrigen Namensbestandteilen vorkommen **Wenn nur *eine* AML-Bibliothek in einem AML-Dokument enthalten ist,** sind folgende Namensbestandteile zu verwenden Prefix | _ | Domain | _ | Type | _ | AMLEdition | _ | LibVersion | .aml • Für *Prefix* und *Type* gelten dieselben Regeln wie in 1.3.1. • Die *Domain* entspricht A_NKL_04 aus Abschnitt 1.3.1. Abweichend davon ist in der Namensgebung der Domain die Angabe ggf. identifizierter Subdomains verpflichtend. • Die *AMLEdition* identifiziert den zugrundeliegenden AutomationML Standard enthalten. ○ AML Edition 1: „AMLEd1“ ○ AML Edition 2: „AMLEd2“ • Die *Libversion* entspricht der Version der Bibliothek, Format a.b.c. • Die Dateierweiterung ist „.aml“. • **Beispiele:** AutomationML_ComponentBase_InterfaceClassLib_AMLEd2_1.1.0.aml	Damit kann man unterschiedliche Versionen von Bibliotheken schon am Dateinamen unterscheiden. Ansonsten könnten die Bibliotheken einfach unbemerkt ausgetauscht werden.	**Pflicht**
A_NKD_02	**Wenn *mehrere* Bibliotheken in einem AML-File enthalten sind,** • sind folgende Namensbestandteile zu verwenden Prefix | _ | Domain | _ | Libraries | _ | AMLEdition | _ | Release-Identifier | .aml • Für *Prefix* gelten dieselben Regeln wie in 1.3.1 beschrieben. • Die *Domain* entspricht A_NKL_04 aus Abschnitt 1.3.1. Abweichend davon ist in der Namensgebung der Domain die Angabe ggf. identifizierter Subdomains verpflichtend. • Anschließend folgt der String „Libraries“, • Die AMLEdition identifiziert den zugrundeliegenden AutomationML Standard. ○ AML Edition 1: „AMLEd1“ ○ AML Edition 2: „AMLEd2“ • Der *Release-Identifier* identifiziert die Version der Bibliothekskollektion, sie ist identisch zur *CAEX OriginVersion* in der zugehörigen *SourceDocumentInformation*. Dieser Release-Identifier wird von der Arbeitsgruppe definiert, die die Bibliothek entwickelt. • Beispiele: ○ AutomationML_ARAPC_Libraries_AMLEd1_1.3.0.aml ○ AutomationML_Part6WPCompo_Libraries_AMLEd1_1.1.0.aml ○ AutomationML_Base.Geometry_Libraries_AMLEd2_2.12.0.aml	Damit kann man unterschiedliche Versionen von Bibliotheken schon am Dateinamen unterscheiden. Ansonsten könnten die Bibliotheken einfach unbemerkt ausgetauscht werden.	**Pflicht**
A_NKD_03	**Pflicht**: Dateinamen von AML Standardbibliotheks-Dateien • Die Namensgebung jedes Bestandteiles erfolgt mit PascalCasing. • Die einzelnen Namensbestandteile dürfen nur die Buchstaben [a-z], [A-Z], [-] sowie [0-9] enthalten, keine Leerzeichen. **Optional**: Bei nutzerdefinierten Bibliotheken ist das eine Empfehlung.	Verbesserte Lesbarkeit	**Pflicht** für AML Standardbibliotheken **Empfehlung** für nutzerdefinierte Bibliotheken

1.3.3 Namenskonventionen für Klassen

ID	Anforderung	Motivation	Priorität
A_NKK_01	**Pflicht: Namen von Klassen (nicht Attributtypen)** • Klassennamen werden via PascalCasing benannt.	Prüfbarkeit sicherstellen	**Pflicht**
A_NKK_02	**Empfehlung:** Klassen sollen ihren Typ nicht im Namen tragen. Der Typ ist softwaretechnisch bekannt, auch die Bibliotheken gibt den Typ bekannt. • Richtig: Roboter • Falsch: Roboter_SUC • Falsch: Roboter_SystemUnitClass	Vorteil: Der Typ ist eindeutig erkennbar.	**Empfehlung**

1.3.4 Namenskonventionen für Attribute und Attributtypen

ID	Anforderung	Motivation	Priorität
A_NKA_01	**Namen von Attributen** **Empfehlung**: Namen von Attributen oder Attributtypen sollen mittels CamelCase benannt werden. • referenceType, speedValue	Übliche Coding-Guidelines	**Empfehlung**
A_NKA_02	**Namen von Attribut-Datentypen** • **Pflicht**: Alle Attributdatentypen sollen benannt werden wie definiert in: http://www.w3.org/TR/xmlschema-2/#built-in-datatypes • z.B. „xs:double“	Kompatibilität und Verbreitung	Pflicht
A_NKA_03	**Benennung von Units** **Empfehlung**: Da ein Xml-Dokument keine hochgestellten Zeichen wie in m^2 enthalten kann, sollte stattdessen die Ersetzung m^2 verwendet werden. Da wir für ein internationales Publikum schreiben, sollten die Werte und Einheiten von Attributen im metrischen System (SI) ausgedrückt werden. Als Referenz für die Definition von SI-Einheiten und -Symbolen soll die https://www.nist.gov/pml/owm/metric-si/si-units verwendet werden. **Pflicht**: die Zeichen */+-^ sind für Rechenoperationen reserviert.	Tool- und sprachunabhängige Lesbarkeit	**Empfehlung**
A_NKA_04	**Notation zusammengesetzter Units** **Pflicht:** die Trennung zwischen Zähler und Nenner kann durch „/“ eingeleitet werden. Für mehrdeutige Ausdrücke sind Klammern zu verwenden, z.B. kg*m/s^2, kg*m/s^(-2), J/(A*s)	Toolunabhängige Lesbarkeit	**Pflicht**

1.4 Konventionen zur Strukturierung von Bibliotheken

ID	Anforderung	Motivation	Priorität
A_S_01	**Empfehlung**: Es wird empfohlen, Klassen in Bibliotheken vorzugsweise als flache Listenstruktur zu modellieren und die Vererbungshierarchie nicht künstlich abzubilden. Stattdessen kann die Vererbungshierarchie durch den AML Editor als „View“ generiert werden.	Der Klassenbaum ist nur eine Sicht auf die Bibliothek. Nutzer verwechseln oft die Bibliotheks-Hierarchie mit der Klassenhierarchie.	**Empfehlung**
A_S_02	**Empfehlung**: Wenn, entgegen der Empfehlung A_S_01, an einer Klasse eine Kindklasse erzeugt, soll die Kindklasse standardmäßig eine Vererbungsbeziehung zur Elternklasse erhalten, indem sie die Elternklasse referenziert.	Vereinfacht das Verständnis für den Benutzer. Tipp: Besser in flachen Listen modellieren.	**Empfehlung**
A_S_03	**Pflicht**: Alle von AML Arbeitsgruppen herausgegebene Bibliotheken dürfen nur ihre eigenen Bibliotheken enthalten. Referenzierte Fremdbibliotheken müssen über CAEX ExternalReferences referenziert und in separaten Dateien gehalten werden. **Empfehlung**: Auch für nutzerdefinierte AML Bibliotheken wird empfohlen, diese in separaten Dateien zu speichern und in ihren Projekten durch Referenzierung zu integrieren.	Das verhindert, das in Kopien dieser Standard-Bibliotheken Modifikationen vorgenommen werden. Es verkleinert auch die Dateien.	**Pflicht** für AML Standardbibliotheken **Empfehlung** für nutzerdefinierte Bibliotheken

1.5 Konventionen für die Modellierung von Bibliotheken, Klassen und Attributtypen

ID	Anforderung	Motivation	Priorität
A_AE_ME_01	**Empfehlung**: Alle Attributtypen sollen direkt oder indirekt eine semantische Referenz auf vorhandene Dictionaries (CC Dictionary, semantische Standards, z.B. andere AML Klassen), **soweit verfügbar.**	Automatisierte Interpretierbarkeit von Merkmalen	**Empfehlung**
A_AE_ME_02	**Pflicht**: CAEX 3.0: Alle in Rollenklassen und Interfaceklassen verwendeten Attribute müssen typisiert sein. Alle Attribute müssen aus einem AML-Attributtyp abgeleitet sein.	untypisierte Attribute werden vermieden	**Pflicht**
A_AE_ME_03	**Empfehlung**: Attributtypen: sollten immer beinhalten: • Description (Sprache: bei AML AG immer englisch) • Datentyp	Verbessert die Menschenlesbarkeit und Typsicherheit. Wer geerbte Werte ändern will, muss das Attribut überschreiben.	**Empfehlung**
A_AE_ME_04	**Pflicht**: • Wenn der Datentyp eines Attributes „xs:string“ ist oder das Attribut ein Strukturattribut wie Listenattribute ist, muss die Einheit leer bleiben und wird auch nicht geprüft. • Wenn der Datentyp ein Zahlenwert ist, z.B. xs:int, xs:float, usw., ist die Einheit immer anzugeben, soweit verfügbar.	Die Fehlersuche wird automatisierbar.	**Pflicht**
A_AE_ME_05	**Pflicht** für AML Standardbibliotheken, **Empfehlung** für nutzerdefinierte Bibliotheken: AML Bibliotheken, Rollenklassen, Interfaceklassen, SystemUnit-Klassen und Attributtypen belegen folgende Felder mit sinnvollem Inhalt: • Description (Sprache: bei AML Arbeitsgruppen englisch) • Version (siehe A_V_01)	Transparenz	**Pflicht** für AML Standardbibliotheken **Empfehlung** für nutzerdefinierte Bibliotheken
A_AE_ME_06	**Empfehlung**: Es sollen nur AML Standard-Bibliotheken referenziert werden, die signiert sind. **Empfehlung**: während der Entwicklung einer Bibliothek sollte das toleriert werden, in finalen Bibliotheken hingegen sollte es nicht toleriert werden.	Unsignierte Bibliotheken sind potenziell manipuliert. Hinweis: Auch Beta-Versionen können signiert werden.	**Empfehlung**
A_AE_ME_07	**Pflicht:** Alle Bibliotheken, Klassen und Attributtypen erhalten eine global eindeutige ID.	Voraussetzung für die Einführung von Referenzen via IDs statt Pfade. Voraussetzung für die AMLID.	**Pflicht**

1.6 Konventionen für Erweiterungsbibliotheken (Extensions)

1.6.1 Namenskonventionen für Erweiterungsbibliotheken

ID	Anforderung	Motivation	Priorität
A_NKE_01	Wenn eine Arbeitsgruppe eine Standardbibliothek, im Folgenden als Mutterbibliothek bezeichnet, erweitert, muss dies auf eine der folgenden beiden Varianten erfolgen. Die gewählte Variante wird in der Arbeitsgruppe festgelegt. **Variante 1:** Wenn eine Standardbibliothek modifiziert wird, veröffentlicht die Arbeitsgruppe eine neue Version der Bibliothek, z.B.: • Mutterbibliothek: AutomationML_Subsea_RoleClassLib (in Version 1.0.0) • Neue Version: AutomationML_Subsea_RoleClassLib (in Version 1.1.0) In Variante 1 gelten die Konventionen aus Abschnitt 1.3. **Variante 2:** Wenn eine Standardbibliothek um Klassen ergänzt werden soll und die ursprüngliche Bibliothek weiterhin alleinstehend nutzbar ist, können diese zusätzlichen Klassen in einer separaten Bibliothek mit eigener Versionierung modelliert werden. Diese Erweiterungsbibliothek existiert unabhängig von seiner Mutterbibliothek und wird als ***Extension*** bezeichnet. • Eine Extension ergänzt die Mutterbibliothek um weitere Elemente, bildet die Mutterbibliothek aber nicht erneut ab. • Eine Mutterbibliothek kann beliebig viele Extensions haben. • Extensions können selbst unabhängig versioniert werden und müssen nicht der Versionsnummer der Mutterbibliothek folgen. • Eine Extension gehört immer zu einer konkreten Mutterbibliotheksversion. Softwarepattern: „Plugin-Pattern“	Der Vorteil einer Extension liegt darin, dass bei Ergänzung die Basisbibliothek nicht erneut veröffentlicht / dokumentiert / signiert usw. werden muss.	**Pflicht für Variante 2**

A_NKE_02	Für Bibliotheksnamen der Erweiterungsbibliotheken gelten folgende Konventionen:
	LibName | _ | ExtPrefix | „Extension“
	• *LibName* entspricht dem Namen der Mutterbibliothek
	• *ExtPrefix* identifiziert den Eigentümer oder die Domäne der Erweiterung.
	• Der *ExtPrefix* kann bei Bedarf durch einen vorangestellten SubPrefix ergänzt werden.
	  ○ Die Syntax der Unterteilung ist *SubPrefix+MainPrefix*.
	  ○ SubPrefixe sind weiter unterteilbar.
	  ○ Der MainPrefix steht am Ende des ExtPrefix.
	  ○ Ein Separator ist nicht erforderlich, ein „.“ empfehlenswert.
	• *„Extension“* steht am Ende des Bibliotheksnamens und identifiziert, dass es sich um eine Erweiterung einer bestehenden Bibliothek handelt.
	• Der Suffix „Extension“ ist nur für Erweiterungsbibliotheken erlaubt.
	Beispiele für die Namenskonventionen für Extensions
	• Name der Mutterbibliothek: HSPF_MasterClass_InterfaceClassLib
	• Name einer Extension: HSPF_MasterClass_InterfaceClassLib_GeometryExtension
	• Name einer Extension mit SubPrefix: HSPF_MasterClass_InterfaceClassLib_Robot.GeometryExtension
	• Name einer Extension mit SubPrefixes: HSPF_MasterClass_InterfaceClassLib_DIN.Robot.GeometryExtension
		Einheitliche Benennung der erweiterten Bibliotheken und Intuitive Zuordenbarkeit, woher sie abgeleitet wurden.	**Pflicht**

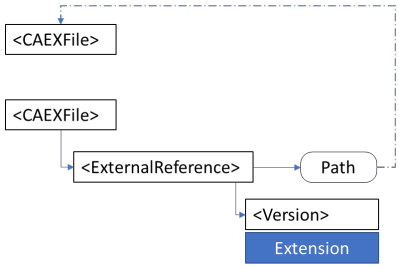
A_NKE_03	Eine Erweiterungsbibliothek darf nur Bibliotheken ergänzen, die in der Mutterbibliothek tatsächlich existieren.		**Pflicht**
A_NKE_04	Eine Erweiterungsbibliothek darf zusätzliche neue Bibliotheken ergänzen, die in keinem Zusammenhang zum Mutterbibliotheks-Dokument stehen und demzufolge keine Extensions sind. Für sie gelten die Namenskonventionen für allgemeine AML-Bibliotheken, siehe 1.3.1.		**Pflicht**
A_NKE_05	Wird eine Erweiterung einer Mutterbibliotheksdatei erzeugt, die eine oder mehrere Mutterbibliotheken enthält, gelten folgende Konventionen: • Ein Erweiterungsbibliotheksdokument kann keine, mehrere oder alle Mutterbibliotheken beliebig oft erweitern und darüber hinaus neue Bibliotheken modellieren. • Der MainPrefix muss für alle Erweiterungen identisch sein.		**Pflicht**

1.6.2 Namenskonventionen für Erweiterungsbibliotheksdateien

A_ENKD_01	Für den Dateinamen eines Extension-Bibliotheks-Dokumentes gilt:
	Pflicht: Dateinamen von AML-Bibliotheks-Extensions müssen einen eindeutigen und im Lebenszyklus der Bibliothek unveränderlichen Namen haben.
	• Ein Dateiname für eine AML-Extension-Bibliothek wird aus folgenden Bestandteilen in der vorgegebenen Reihenfolge zusammengesetzt.
	• Der Separator „_“darf nicht in den übrigen Namensbestandteilen vorkommen.
	Wenn nur *eine* AML-Bibliothek in der Extension-Datei enthalten ist,
	• Es sind folgende Namensbestandteile zu verwenden
	ExtensionLibName | _ | AMLEdition | _ | LibVersion | .aml
	• Der *ExtensionLibName* ist der Name der erweiterten Bibliothek, der in Abschnitt 1.6.1 gebildet wurde.
	• Die *AMLEdition* identifiziert den zugrundeliegenden AutomationML Standard.
	  ○ AML Edition 1: „AMLEd1“
	  ○ AML Edition 2: „AMLEd2“
	• Die *Libversion* entspricht der Version der Bibliothek, Format a.b.c.
	• Die Dateierweiterung ist „.aml“.
	Beispiele
	• Mutterbibliothek: HSPF_MasterClass_InterfaceClassLib
	• Extensionbibliotheksname: HSPF_MasterClass_InterfaceClass_SubseaExtension
	• Dateiname: HSPF_ MasterClass _InterfaceClass_SubseaExtension_AMLEd2_1.0.52.aml
		Damit kann man unterschiedliche Versionen von Bibliotheken schon am Dateinamen unterscheiden.
		Ansonsten könnten die Bibliotheken einfach unbemerkt ausgetauscht werden.
			Pflicht

A_ ENKD_02	**Wenn *mehrere* Bibliotheken in einem Extension-Dokument enthalten sind,** sind folgende Namensbestandteile zu verwenden
	Prefix | _ | Domain | _ | „Libraries“ | _ | ExtPrefix | „Extension“ | _ | AMLEdition | _ | Release-Identifier | .aml
	• Für *Prefix* gelten dieselben Regeln wie in 1.3.1 beschrieben.
	• Die *Domain* identifiziert das Anwendungsgebiet. In der Namensgebung der Domain ist die Angabe ggf. identifizierter Subdomains verpflichtend.
	• „Libraries“ identifiziert die Bibliothek als Kollektion mehrerer Bibliotheken.
	• *ExtPrefix* identifiziert den Eigentümer oder die Domäne der Erweiterung.
	  ○ Enthält der *ExtPrefix* der Extension *SubPrefixes*, wird im Dateinamen nur der MainPrefix verwendet, vgl. 1.6.1.
	• *„Extension“* identifiziert, dass es sich um eine Erweiterung einer bestehenden Bibliothek handelt.
	• Die *AMLEdition* identifiziert den zugrundeliegenden AutomationML Standard enthalten.
	  ○ AML Edition 1: „AMLEd1“
	  ○ AML Edition 2: „AMLEd2“
	• Der *Release-Identifier* identifiziert die Version der Bibliothekskollektion, sie ist identisch zur *CAEX OriginVersion* in der zugehörigen *SourceDocumentInformation*. Dieser Release-Identifier wird von der Arbeitsgruppe definiert, die die Bibliothek entwickelt.
	Beispiele:
	• HSPF_MasterClass_Libraries_RobotExtension_AMLEd2_1.1.1.aml
		Damit kann man unterschiedliche Versionen von Bibliotheken schon am Dateinamen unterscheiden.
		Ansonsten könnten die Bibliotheken einfach unbemerkt ausgetauscht werden.
			Pflicht

1.6.3 Konventionen für die Modellierung von Erweiterungsbibliotheken

A_EKM_01	**Pflicht**: Extensions von Extensions sind nicht erlaubt. Ist eine vorhandene Extension unzureichend, ist eine neue eigenständige Extension zu modellieren oder eine neue Version der bestehenden Extension zu erstellen.	Bessere Koordination von Extensions, Vermeidung von Wildwuchs.	**Pflicht**
A_EKM_02	**Pflicht**: Die Extension-Bibliothek soll sich wie in Abschnitt 2.1 beschrieben im Feld *SourceDocumentInformation* identifizieren.	Maschinenlesbare Nachvollziehbarkeit, welche Bibliothek genau erweitert wird.	**Pflicht**
A_EKM_03	**Pflicht**: Eine Erweiterungsbibliotheksdatei referenziert ihre Mutterbibliothek-Datei mit Hilfe einer CAEX ExternalReference. • Um eine Unterscheidung von normalen CAEX ExternalReferenes zu ermöglichen, müssen *ExtensionReferences* ein Version-Element mit dem Wert „Extension“ (**Fehler! Verweisquelle konnte nicht gefunden werden.**) haben. • Die Mutterbibliothek referenziert ihre Extensions nicht.  Abbildung 1: ExtensionReference	Maschinenlesbare Nachvollziehbarkeit, welche Bibliothek genau erweitert wird.	**Pflicht**
A_EKM_04	**Pflicht:** Jede Extension-Bibliothek referenziert ihre Mutterbibliothek, siehe 1.7.4		
A_EKM_05	**Pflicht:** Extensions dürfen nur Bibliotheken der gleichen AutomationML Edition und der gleichen CAEX-Version erweitern.	Wahrung der Eindeutigkeit der verwendeten AML oder CAEX Version.	**Pflicht**
A_EKM_06	**Pflicht**: Ein AML Dokument, das eine Extension referenziert, muss zugleich auch die Mutterbibliothek referenzieren. Beliebig viele Extensions derselben Mutterbibliothek dürfen referenziert werden.		**Pflicht**
A_EKM_07	**Pflicht:** Eine Extension enthält nur neue Klassen/Typen oder Ableitungen von Klassen/Typen anderer Bibliotheken einschließlich der Mutterbibliothek.		**Pflicht**
A_EKM_08	**Pflicht**: Wird in einem AML Dokument eine Extension referenziert, darf das AML Dokument zugleich keine ältere Version der Extension referenzieren.	Vermeidung des Vermischens von Versionen.	Pflicht

1.7 Konventionen zur Versionierung von Klassen und Bibliotheken

1.7.1 Begriffsbestimmung: Versionierung versus Vererbung

Das Erstellen einer neuen Version einer Klasse und das Erzeugen einer Ableitung einer Klasse sind zwei unterschiedliche Vorgehensweisen in der objektorientierten Programmierung, die verschiedene Zwecke erfüllen und in unterschiedlichen Kontexten angewendet werden. Vererbung und Versionierung sind wichtige Werkzeuge in der objektorientierten Datenmodellierung und werden je nach den spezifischen Anforderungen und Zielen eingesetzt.

Eine *neue Version* einer Klasse oder Bibliothek ist erforderlich, wenn eine bereits veröffentlichte und bereits verwendete Bibliothek oder Klasse verändert werden soll, um Verbesserungen, Bugfixes, Fehlerbehebungen oder neue Funktionalitäten zu modellieren, aber die originale Klasse nicht verändert werden kann oder darf. In diesen Fällen wird eine neue Version der Klasse erzeugt, wobei die Bedeutung der Klasse gleichbleibt.

Beispiel: Aus einem „Einarm-Roboter Version 1.0.0“ wird eine neue Version „Einarmroboter Version 1.1.0“ erstellt. Die alte und neue Version einer Klasse bestehen unabhängig, sind bedeutungstechnisch aber gleich. Im Unterschied zur Vererbung besteht zwischen der alten und neuen Version einer Klasse keine Vererbungsbeziehungen: Änderungen in der alten Version haben keinen Einfluss auf die neue Version. Die neue Version ist zunächst eine Kopie der alten Version und darf im Modellierungsprozess bis zur Finalisierung und Publikation beliebig modifiziert werden. Eine neue Version geht folglich mit einem Bruch der Vererbungsbeziehung einher. Editionsübergreifende Versionsreferenzen sind erlaubt, da sie keine Relevanz für das Datenmodell besitzen, sondern informativ sind und von Tools ausgewertet werden können.

Vererbung sollte hingegen verwendet werden, wenn eine spezialisierte oder erweiterte Variante einer Klasse benötigt wird, die die Funktionalität und Bedeutung der Basis-Klasse nutzt und erweitert. Beispiel: von einer Klasse „Einarm-Roboter Version 1.0.0“ wird eine neue Kindklasse „Zweiarmroboter 1.0.0“ abgeleitet – diese erhält eine neue Bedeutung. Änderungen in der Elternklasse werden in der Kindklasse unmittelbar reflektiert. Editionsübergreifende Vererbung ist deshalb explizit nicht erlaubt.

1.7.2 Begriffe VersionReference und ExtensionReference

Referenziert ein CAEX Dokument ein anderes Dokument, erfolgt dies standardkonform über CAEX ExternalReferences. Wenn ein AutomationML-Dokument (CAEXFile) erstellt wird, das neue Versionen von Bibliotheken und Klassen aus einem anderen Dokument enthält, wird eine wechselseitige Beziehung zwischen diesen beiden Dokumenten ebenfalls über CAEX-ExternalReferences hergestellt.

Wenn diese Beziehungen speziell der Versionierung dienen, fungieren diese ExternalReferences als *VersionReference*. Eine *VersionReference* auf eine neue Version wird auch als *Vorwärtsreferenz* bezeichnet. Eine *VersionReference* auf eine alte Version wird als *Rückwärtsreferenz* bezeichnet. Sowohl Bibliotheken als auch Klassen oder Typen referenzieren sich gegenseitig über Vorwärts- und Rückwärtsreferenzen.

AutomationML-Bibliotheken können durch Extensions erweitert werden. Eine Extension ist eine eigenständige Bibliothek und referenziert ihre Mutterbibliothek über eine *CAEX ExternalReference*. Diese spezielle ExternalReference wird als *ExtensionReference* bezeichnet. Für Extensions sind nur Rückwärtsverweise, aber keine Vorwärtsverweise definiert.

1.7.3 Allgemeine Konventionen für das Modellieren von Versionen

Für die Modellierung von Versionen von Bibliotheken, Klassen oder Typen stellt der AutomationML e.V. folgende Richtlinien auf.

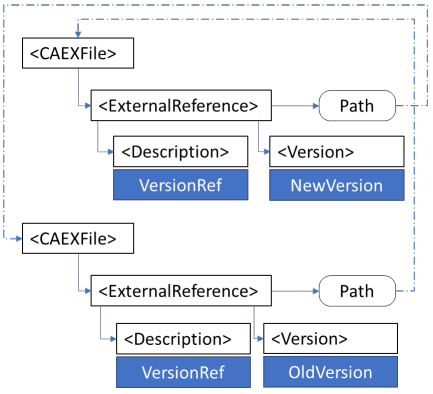
ID	Anforderung	Motivation	Priorität
A_V_01	**Pflicht**: Bibliotheken müssen versioniert werden. Dazu muss das CAEX Attribut *Version* gesetzt werden.	Vorbereitung von nachhaltiger Entwicklung.	Pflicht
A_V_02	**Empfehlung**: Klassen müssen versioniert werden. Dazu muss das CAEX Attribut *Version* gesetzt werden.	Vorbereitung von nachhaltiger Entwicklung.	Empfehlung
A_V_03	**Pflicht**: Die Versionsnummer soll dem Konzept des *semantic versioning* folgen und a.b.c notiert sein (siehe https://semver.org). a=Major Release, bricht die Kompatibilität b=Minor = Ergänzung, rückwärtskompatibel c=Patch = Arbeitsversion rückwärtskompatibel • Nicht abwärtskompatible Bibliotheken zählen a hoch. • abwärtskompatible Bibliotheken zählen b hoch. • Arbeitsversionen zählen c hoch. Für Pre-Release-Versionen können beliebige Pre-Release-Tags an die Versionsnummer angefügt werden, z.B. „-alpha0.1“.	Erleichtert das geführte softwaregestützte Nachvollziehen der Versionshistorie.	Pflicht
A_V_04	**Pflicht**: Wird eine neue Version einer Klasse oder eines Attributtypen erzeugt, muss sie in einer neuen Bibliothek modelliert werden, da ansonsten die Signatur der ursprünglichen Bibliothek ungültig würde. **Beispiel**: Soll innerhalb einer AML Edition eine AML Standardbibliothek modifiziert werden, muss sie in einer neuen Version veröffentlicht werden. Eine Erweiterung existierender Bibliotheken darf nie ohne Auswirkungen auf ihre Version vorgenommen werden.	Bewirkt große Klarheit bei der Referenzierung.	**Pflicht**
A_V_05	**Empfehlung**: Wird eine neue Version einer Klasse, eines Attributtyps oder eine Bibliothek erzeugt, gelten bezüglich der ID folgende Regeln: • Wenn eine Bibliothek, Klasse oder ein Attributtyp inhaltlich bzw. nach ihrem Wesen gleichbleibt, behält es seine ID auch in der neuen Version. • Kleinere redaktionelle Anpassungen, die die Bedeutung und den Anwendungsbereich nicht verändern (z.B. Korrektur eines Tippfehlers), führen nicht zu einer neuen ID. • Wird eine Bibliothek, Klasse oder ein Attributtyp erweitert, z.B. durch Hinzufügen weiterer erlaubter Elemente oder durch Präzisierungen, bleibt die ID ebenfalls bestehen, solange der ursprüngliche Zweck erhalten bleibt. • Erhält eine Bibliothek oder Klasse eine neue Major-Version, bleibt seine ID grundsätzlich erhalten, solange die Bedeutung oder der Anwendungsbereich nicht verändert ist. • Sonderfall: Wird eine Bibliothek, Klasse oder ein Attributtyp so verändert, dass es seinen ursprünglichen Zweck oder seine Bedeutung verliert, wird eine neue ID vergeben. • eine neue ID erfordert immer eine neue Major Version. Ein(e) vollständig neue Bibliothek, Klasse oder Attributtyp erhält eine neue eindeutige ID.	Vereinfacht die Identifizierung. Dies verhindert Missverständnisse und stellt sicher, dass alte und neue Versionen des Merkmals klar unterscheidbar sind.	
A_V_06	**Pflicht**: Wird eine AML Bibliothek von einem alten auf einen neuen AML Standard konvertiert (z.B. von AutomationML Ed. 1 nach Ed. 2), ist die Versionsnummer der neuen Bibliothek unabhängig von der Versionsnummer der ursprünglichen Bibliothek und kann z.B. von vorne beginnen. Aus Gründen der besseren Vergleichbarkeit kann die Versionsnummer von Bibliotheken unterschiedlicher Editionen bei inhaltlicher Gleichheit identisch gewählt werden. Die Versionen beider Bibliotheken können sich jedoch anschließend unabhängig weiterentwickeln, deshalb lässt sich aus der Gleichheit der Bibliotheks-Versionsnummern über AML Editionsgrenzen hinweg nicht rückschließen, dass sie inhaltlich identisch sind. Hinweis: Das Erkennen der	Unterscheidbarkeit der Bibliotheken, auch wenn sie sich inhaltlich nicht unterscheiden.	Pflicht

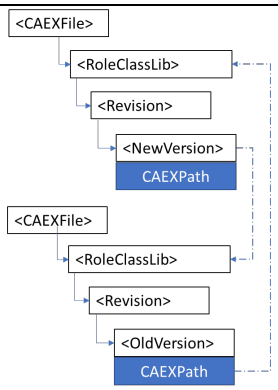
	AML Edition erfolgt nicht über die Versionsnummer, sondern über andere AML Mechanismen.		
A_V_07	**Pflicht**: Eine neue Bibliotheks-Version muss in einem neuen Dokument gespeichert werden und darf sich nicht im selben Dokument befinden wie die ältere Version.	Einheitliche Trennung von Versionen	Pflicht
AV_V_08	Wird eine neue Patch- oder Minor-Version einer Bibliothek erzeugt, dann gilt: • Die neue Bibliothek sei abwärtskompatibel mit der vorigen Version, sie enthält alle Klassen/Typen der vorigen Version unter ihrem unveränderten Pfad, d.h. alle Namen und hierarchische Positionen aller bestehenden Bibliothekselemente bleiben unverändert. • Die neue Bibliothek erhält eine neue Versionsnummer mit inkrementiertem b oder c (siehe A_V_03). • Unveränderte Klassen inkrementieren ihre Patch Version c. • Modifizierte abwärtskompatible Klassen erhalten eine höhere Versionsnummer und zählen b oder c hoch. • Ergänzungen der Bibliothek sind erlaubt, sofern sie die Abwärtskompatibilität nicht brechen, z.B. neue Klassen/Typen oder Ergänzungen an bestehenden Klassen/Typen. • Das Löschen, Umbenennen oder Umpositionieren von Klassen/Typen oder von ihren Eigenschaften ist in abwärtskompatiblen Bibliotheken nicht erlaubt. • Ist ein Bibliothekselement nicht abwärtskompatibel, ist die gesamte Bibliothek nicht mehr abwärtskompatibel. Dann ist eine neue Major-Version der Bibliothek erforderlich, siehe AV_V_09.	**Ziel:** Eine Patch- oder Minor-Version einer Bibliothek ist austauschbar mit ihrer Vorversion. Alle Klassen oder Typen sind über ihren Pfad weiterhin findbar. Änderungen im Code der verarbeitenden Software sind gering oder nicht erforderlich.	Pflicht
AV_V_09	Wird eine neue Major-Version einer Bibliothek erzeugt, dann gilt: • Eine neue Major-Version einer Bibliothek bricht die Abwärtskompatibilität zur vorigen Version. • Die neue Bibliothek erhält eine neue Versionsnummer mit inkrementiertem a. • Unveränderte Klassen inkrementieren ihre Patch Version c. • Abwärtskompatible modifizierte Klassen erhalten eine höhere Versionsnummer und zählen b oder c hoch. • Nicht abwärtskompatible modifizierte Klassen erhalten eine höhere Versionsnummer und zählen a hoch. Sie dürfen umbenannt, umpositioniert und beliebig verändert werden. • Nicht mehr benötigte Bibliothekselement dürfen entfernt werden. Alternativ kann ihr CAEX ChangeMode mit dem Wert „Deleted“ gesetzt werden.	Eine neue Major-Version einer Bibliothek erfordert Anpassungen im Code der verarbeitenden Software.	

1.7.4 Allgemeine Konventionen zur gegenseitigen Referenzierung von Versionen

A_RV_01	**Pflicht**: Wird eine normale Bibliothek, ein Typ oder eine Klasse in einer neuen Version modelliert, sollen die alte und neue Version sich gegenseitig referenzieren. Dazu verweist die alte Version im Feld Revision/newVersion auf die neue Version, wohingegen die neue Version im Feld Revision/oldVersion auf die alte Version referenziert. Wenn eine neue Version mit einer vorigen bricht, d.h. inhaltlich über eine Änderung der Major Version hinausgeht, wird sie als neue unabhängige Klasse modelliert und muss nicht auf die vorige Version zeigen. Wird eine Bibliothek von einer alten AML Edition in eine AML Edition konvertiert, sollen die alte und neue Bibliothek ebenfalls aufeinander verweisen. Editionsübergreifende Versionsreferenzen sind explizit erlaubt, da sie keine Relevanz für das Datenmodell besitzen, sondern informativ sind. Ihre Auswertung ist eine Softwarefunktion außerhalb von AutomationML.	Erleichtert das geführte softwaregestützte Nachvollziehen der Versionshistorie sowie das Verteilen, Erkennen, Einpflegen bzw. Informieren über neue Versionen.	Pflicht
A_RV_02	**Pflicht**: Die Syntax der gegenseitigen Referenzen einer alten und neuen Version in den CAEX Feldern *newVersion* und *oldVersion* lautet „<alias>@<path>“. • „<alias>“ repräsentiert eine CAEX ExternalReference, • „<path>“ repräsentiert den Pfad zum Ziel.		Pflicht
A_RV_03	**Pflicht**: Wird eine Erweiterungsbibliothek (Extension) modelliert, muss jede erweiterte Bibliothek ihre Mutterbibliothek referenzieren. Dazu verweist die erweiterte Bibliothek im Feld Revision/oldVersion auf Mutterbibliothek. Die Mutterlib referenziert ihre Extensions hingegen nicht.		Pflicht

1.7.5 Spezielle Konventionen zur Modellierung von Versionsreferenzen

A_VR_01	**Pflicht**: Die Referenz zu einem anderen AML Dokument, das eine anderen Version einer Bibliothek und/oder Klasse enthält, wird mit Hilfe einer CAEX ExternalReference modelliert. • Um eine *VersionReference* von einer normalen ExternalReference zu unterscheiden, erhalten *VersionReferences* ein Description-Element mit dem Wert „VersionRef“. • Zusätzlich wird ein Label hinzugefügt, das festlegt, welche Referenz auf die neue Version und welche auf die alte Version verweist. Zu diesem Zweck wird ein CAEX Version-Element mit dem Wert „NewVersion“ oder „OldVersion“ hinzugefügt, siehe Abbildung 3. • Im Gegensatz zu einer normalen ExternalReference können mit einer VersionReference AutomationML-Dokumente verschiedener AutomationML-Editionen oder verschiedener CAEX-Versionen in Beziehung gesetzt werden.  *Abbildung 2: VersionReference Relation*	Die Markierung von CAEX ExternalReferences mit dem Feld „Description = VersioRef“ weist darauf hin, dass diese Datei nicht physisch vorhanden sein muss sondern rein informativ ist. Dies verhindert, dass eine Bibliothek alle vorherigen Versionen vollständig mitführen muss.	
A_VR_02	**Pflicht**: Ein AML Dokument darf nicht verschiedene Versionen derselben Bibliothek referenzieren. Wird eine Bibliothek für die Modellierung verwendet (z.B. für das Bilden von Instanzen oder Ableitungen), darf keine ältere Version dieser Bibliothek ebenfalls verwendet werden. Eine Referenz auf ältere Klassen ist nur in den Revision-Objekten in den Elementen NewVersion und OldVersion möglich, wie in der Abbildung 3 gezeigt.	Auf diese Weise wird das Vermischen verschiedener Versionen und das ungewünschte Mitschleppen der	

	 Abbildung 3: Erlaubte Relationen zwischen Dokumentversionen **Beispiel:** Ein AML Dokument verweist auf eine Bibliothek Version 1.1.0 und instanziiert Klassen aus dieser Bibliothek. Die Bibliothek 1.1.0 referenziert ihrerseits über eine VersionReference ihre frühere Version 1.0.0. Das AML Dokument darf dann die Bibliothek 1.0.0 nicht referenzieren oder verwenden. Die Versionsreferenz hat nur informativen Charakter, Version 1.0.0 muss physisch nicht präsent sein.	Versionshistorie verhindert.	

2 Konventionen für das Publizieren und die Verwendung publizierter AML Bibliotheken, Klassen und Typen

2.1 Selbstidentifikation und Signierung von Bibliotheken und Extensions

<table>
<tr><th>ID</th><th>Anforderung</th><th>Motivation</th><th>Priorität</th></tr>
<tr>
<td>A_NM_01</td>
<td>
Pflicht: Eine publizierte AutomationML Bibliotheksdatei muss sich selbst im CAEX Dokument identifizieren, d.h. sie soll intern einen Hinweis auf seine digitale Originalquelle speichern.

Wenn in der Bibliotheksdatei nur <u>eine</u> Bibliothek enthalten ist, müssen die folgenden Felder der CAEX SourceDocumentInformation ausgefüllt werden, um diese Bibliothek zu identifizieren.
- OriginID: eindeutiger Identifier für die Bibliothek – dieses Feld wird genutzt, um der Bibliothek eine ID zu geben, die sie bisher nicht hatte
- OriginName: eindeutiger Name für die Bibliothek
- OriginVersion: Versionsnummer der Bibliothek
- LastWritingDateTime: Datum des Releases
- OriginVendor: Name des Erzeugers oder der Erzeugergruppe
- OriginVendorURL: Name der Bibliotheksdatei

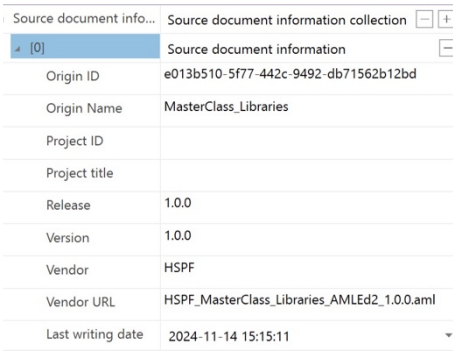
Abbildung 4 zeigt dies an einem Beispiel.


Abbildung 4: SourceDocumentInformation einer Bibliothek

Wenn in der Bibliotheksdatei <u>mehrere zusammengehörige</u> Bibliotheken enthalten sind, z.B. eine oder mehrere Attribut-, Rollen- und Interfaceklassenbibliotheken, bilden sie eine Bibliotheks-Kollektion. Um diese Kollektion insgesamt zu identifizieren, muss für die Kollektion die folgenden Felder der CAEX SourceDocumentInformation ausgefüllt werden.
- OriginID: eindeutiger Identifier für die Bibliothek – dieses Feld wird genutzt, um der Bibliothek eine ID zu geben, die sie bisher nicht hatte.
- OriginName: eindeutiger Name für die Kollektion
- OriginVersion: Versionsnummer der Kollektion
- LastWritingDateTime: Datum der Publikation
- OriginVendor: Name des Erzeugers oder der Erzeugergruppe
- OriginVendorURL: Name der Bibliotheksdatei.

Das Speichern mehrerer unabhängiger Kollektionen von Bibliotheken in einem gemeinsamen AML Dokument ist technisch möglich, aber nicht erlaubt.
</td>
<td>Ziel ist, die Bibliothek einer Erzeugergruppe zuzuordnen und die Version der Release zu definieren.</td>
<td>Pflicht</td>
</tr>
</table>

A_NM_02	**Pflicht**: Fremd-AML Bibliotheken sind stets durch Referenzierung zu verwenden, dadurch bleiben alle Bibliotheken separat, geschützt und können unabhängig ersetzt werden.	Verbesserung der Nachvollziehbarkeit und Downloadbarkeit der Bibliotheken.	**Pflicht**
A_NM_03	**Pflicht**: AML Standardbibliotheken werden signiert. **Empfehlung**: Nutzerdefinierte Bibliotheken werden signiert. Das bedeutet, dass umgekehrt jede Änderung einer Bibliothek zum Bruch der Signatur führt. Daraus leitet sich ab, dass Bibliotheken nach Signatur und Veröffentlichung in ihrer Version nicht mehr unbemerkt geändert werden können.	Die Signatur schützt eine Bibliothek vor ungewollter Modifikation oder ermöglicht ein Ausrollen von Informationen über neue Versionen.	**Pflicht** für AML Standardbibliotheken **Empfehlung** für nutzerdefinierte Bibliotheken

2.2 Empfehlungen zur online-Verwendung von Bibliotheken, Klassen und Typen

ID	Anforderung	Motivation	Priorität
A_OM_01	**Pflicht**: AML Bibliotheken sind, sofern sie einen Publikationsprozess durchlaufen haben, online verfügbar, online findbar und vom AML Editor online zugreifbar. Dies erfordert Online-Zugang.	Online-Zugang ist Voraussetzung für alle nachfolgenden Features.	**Pflicht**
A_OM_02	**Empfehlung:** AML Attributtypen sollen global referenzierbar werden. • Um einen Attributtyp semantisch zu referenzieren, wird in die in das Feld *CorrespondingAttributePath* der CAEX RefSemantic des Attributtyps der String „AMLID:<guid>“ eingetragen. • Eine semantische Referenz eines CAEX-Attributs auf ein AML-Standardattribut besteht aus einem Präfix "AMLID", einem Trennzeichen ":" und einer GUID, die im Feld semanticRef gespeichert wird. • Mehrere semantische Referenzen sind erlaubt. AML Attributtype-Libs werden damit selbst zu einem referenzierbaren semantischen Standard, die wiederum selbst auf bestehende semantische Standards referenzieren können, auch mehrere.	AML Semantik wird damit selbst referenzierbar. AML kann bestehende heterogene semantische Standards zusammenführen und harmonisieren helfen.	**Empfehlung**
A_OM_03	**Empfehlung**: Wird ein neuer Attributtyp definiert und ist dieser bereits vorhanden, soll er nicht erneut modelliert, sondern wiederverwendet werden, um eine Remodellierung von Attributtypen zu vermeiden.	Auf diese Weise kann eine nutzerdefinierte Bibliothek aus mehreren Quellen zusammengesetzt werden ohne Remodellierung bereits bestehender Typen.	**Empfehlung**
A_OM_04	**Empfehlung**: Wird eine neue Klasse definiert und sie ist bereits vorhanden und online verfügbar, soll sie nicht erneut modelliert, sondern wiederverwendet werden.	Vermeidung doppelter Modellierung. Wiederverwendung von Modellen.	**Empfehlung**

2.3 Konventionen zum Speichern von referenzierten Bibliotheken und AML-Dokumenten

<table>
<tr><th>ID</th><th>Anforderung</th><th>Motivation</th><th>Priorität</th></tr>
<tr><td>A_S_01</td><td>Pflicht: Werden AML-Dokumente auf einem Datenträger gespeichert, müssen Kopien verwendeter Dritt-Bibliotheken heruntergeladen und in einem lokalen Unterordner für die offline-Verwendung vorgehalten werden. Der Name des Unterordners lautet „_AMLExternalLibraries“. Darin befinden sich Offline-Kopien der verwendeten Bibliotheken direkt im Dokumentenordner.
Diese Bibliotheken sind im Hauptdokument des eigenen AML-Dokuments über einen geeigneten Alias zu referenzieren. Die Referenzierung erfolgt über einen Relativpfad.
Hinweis: Das Versenden aller zusammengehörigen Dokumente kann mittels Containerformat AMLX erfolgen. Fehlende Bibliotheken können vom AML Editor jedoch selbständig heruntergeladen werden.</td><td>Alle Fremdbibliotheken liegen im Unterordner „_AMLExternal-Libraries“. Manipulationsschutz, kleinere Dateien, Vermeidung von Mehrfachladen von widersprüchlichen Bibliotheken.</td><td>Pflicht</td></tr>
<tr><td>A_S_02</td><td>Pflicht: Für die Referenzierung externer Bibliotheken aus dem eigenen AML Dokument muss je ein Alias benannt werden. Hierfür gelten folgende Regeln:
Ein Alias muss innerhalb des Dokuments eindeutig sein
Umsetzungsempfehlung: Dies kann dadurch erreicht werden, indem der Alias dem Namen der Zielbibliothek (nicht der Datei) entspricht.
Umsetzungsempfehlung: Wird eine kompaktere Form gewünscht, ist dies erlaubt, solange die Eindeutigkeit gewahrt ist.</td><td>Menschenlesbarkeit</td><td>Pflicht</td></tr>
<tr><td>A_S_03</td><td>Pflicht: Wird eine lokale Bibliothek in ein AML Dokument integriert, muss der Pfad relativ zum Dokumentenordner auf die Libs zeigen.</td><td>Nachvollziehbarkeit</td><td>Pflicht</td></tr>
<tr><td>A_S_04</td><td>Pflicht: wird eine publizierte AML- oder Dritt-Bibliothek von einem Server in einer offline-Kopie heruntergeladen, wird der konkrete Pfad zum AML Server als CAEX ExternalReference in diese Kopie eingetragen.
 

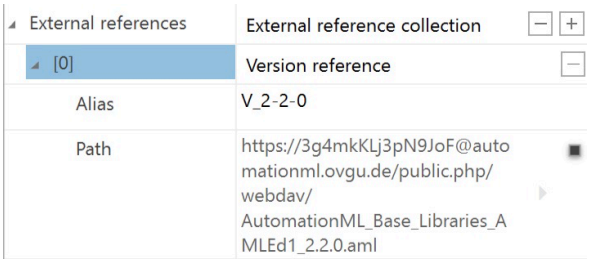
Abbildung 5: Die Offline-Kopie einer AML Bibliothek wird mit via CAEX ExternalReference um einen konkreten Pfad zu seinem Original ergänzt.
Beispiel: Abbildung 6 zeigt die Selbstreferenzierung einer offline-Kopie einer Bibliothek.
 

Abbildung 6: Selbstreferenzierung der offline-Kopie einer Bibliothek</td><td>Der Pfad zum Originalbibliothek erlaubt den Zugriff auf die Bibliothek, z.B. wenn die Signatur verletzt wurde. Ändert sich der Server, muss die externe Referenz aktualisiert werden.</td><td></td></tr>
</table>

Die Umsetzung wird in Abbildung 7 illustriert.

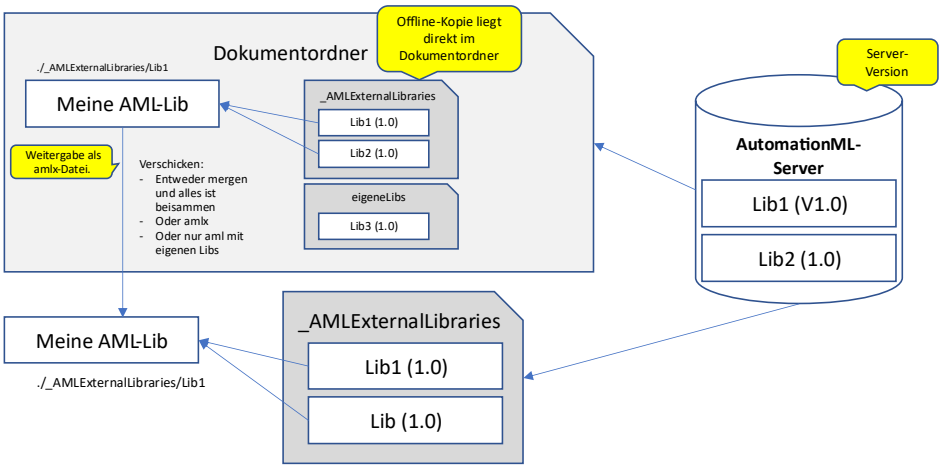


Abbildung 7: Verteilte Dokumentarchitektur eines AML Dokuments, z.B. einer AML Bibliothek

2.4 Rechteverwaltung von Bibliotheken

ID	Anforderung	Motivation	Priorität
A_R_01	Masterbibliotheken dürfen nur von befugten Personen verändert werden. Das Rechtesystem dafür ist zu definieren.	Manipulationsschutz IP Protection Signierung	**Pflicht**
A_R_02	Bibliotheken müssen vor ungewollte Modifikation geschützt werden.	Manipulationsschutz	**Pflicht**