



# <AutomationML/>

**The Glue for Seamless  
Automation Engineering**

**Best Practice Recommendations:  
AutomationML Container**

**Document Identifier: BPR Container, V 1.0.0**

**State: October 2017**

## Table of contents

Table of contents .....	2
List of figures .....	2
Preface .....	3
1 Motivation for the realization of AutomationML as Container .....	3
2 Realization.....	3
2.1 relationshipTypes.....	3
2.2 Structure of an AutomationML Container .....	5
2.3 Referencing into an AutomationML Container.....	5
3 Generate an AutomationML Container .....	6
4 Example implementation.....	6
5 Code examples .....	7
6 References .....	7

## List of figures

<i>Figure 1 – Structure of an AutomationML Container .....</i>	<i>5</i>
<i>Figure 2 – Two Steps for the generation of an AutomationML Container .....</i>	<i>6</i>
<i>Figure 3 – create new AML Container file .....</i>	<i>7</i>
<i>Figure 4 – extract an AML Container.....</i>	<i>7</i>

## Preface

AutomationML provides the basis for an efficient data exchange within the engineering process of production systems. The AutomationML standard series IEC 62714 "Engineering data exchange format for use in industrial automation systems engineering" already contains many use cases and guidelines of how system engineering information is modelled.

In order to specify these definitions with examples, to apply them to specific use cases, and to facilitate the first steps with AutomationML, specific issues for the modelling of data in AutomationML are illustrated in Best Practise Recommendations (BPR).

In addition, the BPR shall provide a consistent realisation for specific use cases and shall thus, complement the AutomationML standard documents.

## 1 Motivation for the realization of AutomationML as Container

AutomationML is realized as a distributed document format, i.e. a project can be stored in different files of different formats. The files can be available locally or even centrally. Thus, they can be stored, for example, in a project share or in the internet. For that, it is important to know the root file that contains the project entry point. Additionally, it is also important to know which file is a part of the project in the data transfer.

To cope with these challenges, a complete AutomationML project can be stored by using an AutomationML Container. In doing so, all relevant files resp. all files, that are not centrally available, can be stored in an ECMA OPC archive as one package.

On the one hand, this enables the different involved software tools to find the project's entry point and on the other hand this enables the involved experts to transfer complete projects. As a side effect, the file size can be reduced significantly due to the compression as an archive, since XML based files enable a high packaging density.

## 2 Realization

Two types of AutomationML Containers are allowed: completely standalone, offline (self containing /pack'n'go) AutomationML Containers and AutomationML Containers with project public links (non-self containing /interlinked containers).

For completely standalone, offline AutomationML Containers, all files belonging to the AutomationML project have to be in the AutomationML Container. Only relative, local links are allowed between the files.

For AutomationML Containers with project public links, links with URIs are allowed, additionally, which refer to locations which are public available for all the different involved software tools or experts.

AutomationML Containers can include further AutomationML Containers.

The referencing to these elements is described in clause 2.3. AutomationML Container files have the file extension ".amlx".

### 2.1 relationshipTypes

Relationships within an OPC Container file are represented by so called relationshipTypes.

The following relationshipTypes shall be used for an AutomationML Container:

#### ■ Root AML File

A Root AML file represents an AutomationML file that serves as the entry point for an AutomationML Container.

relationshipType: <http://schemas.automationml.org/container/relationship/RootDocument>

Mime type: "model/vnd.automationml+xml"

#### ■ Library AML File

A Library AML file is an AutomationML library file that usually contains only RoleClassLibs, InterfaceClassLibs, and SystemUnitClassLibs. Like the Root AML file, a library file is always defined as an entry point for an AutomationML Container. In this manner, it is possible to create AutomationML Containers which only contain library files.

relationshipType: <http://schemas.automationml.org/container/relationship/Library>

MIME type: "model/vnd.automationml+xml"

#### ■ COLLADA File

relationshipType: <http://schemas.automationml.org/container/relationship/Collada>

MIME type: "model/vnd.collada+xml"

#### ■ PLCopen XML File

relationshipType: <http://schemas.automationml.org/container/relationship/PLCOpenXML>

MIME type: "model/vnd.plcopen+xml"

#### ■ Any Content

relationshipType: <http://schemas.automationml.org/container/relationship/AnyContent>

MIME type: "application/x-any" or user defined

#### ■ CAEX Schema

relationshipType: <http://schemas.automationml.org/container/relationship/CAEXSchema>

MIME type: "text/xml"

#### ■ COLLADA Schema

relationshipType: <http://schemas.automationml.org/container/relationship/ColladaSchema>

MIME type: "text/xml"

#### ■ PLCopen XML Schema

relationshipType: <http://schemas.automationml.org/container/relationship/PLCOpenXMLSchema>

MIME type: "text/xml"

## 2.2 Structure of an AutomationML Container

The following figure shows the structure of an AutomationML Container.

### AutomationML Container

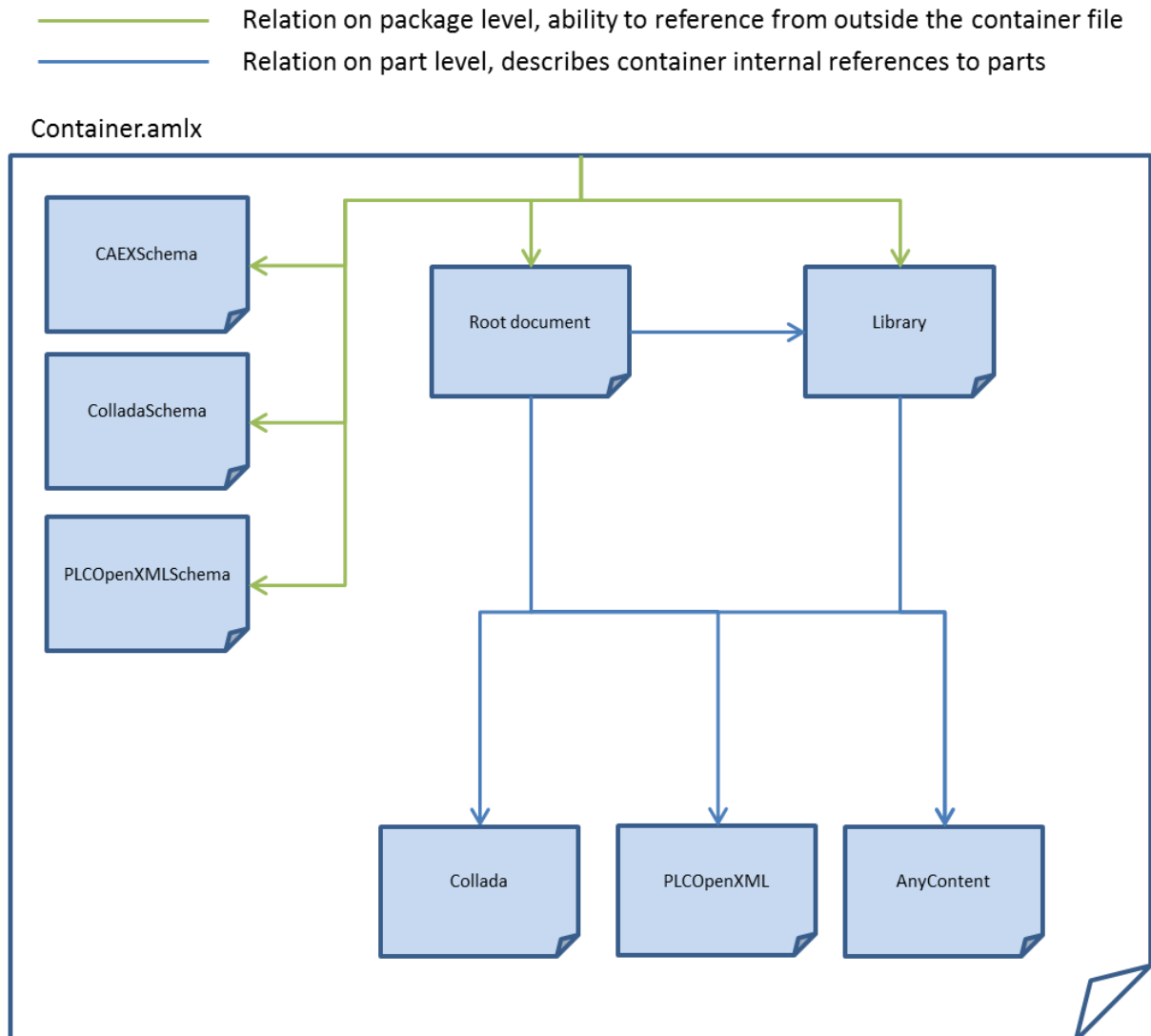


Figure 1 – Structure of an AutomationML Container

## 2.3 Referencing into an AutomationML Container

For referencing an OPC Container via URI references the “pack” scheme is used. See:

- <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-376,%20Fourth%20Edition,%20Part%202%20-%20Open%20Packaging%20Conventions.zip>
- [http://standards.iso.org/ittf/PubliclyAvailableStandards/c061796 ISO IEC 29500-2 2012.zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c061796%20ISO%20IEC%2029500-2%202012.zip)

Examples for URIs:

- `pack://http:,,www.automationml.org,container.amlx/root.aml`
- `pack://file:,,c:,temp,container.amlx/root.aml`

### 3 Generate an AutomationML Container

The following figure shows the two steps necessary to generate an AutomationML Container.

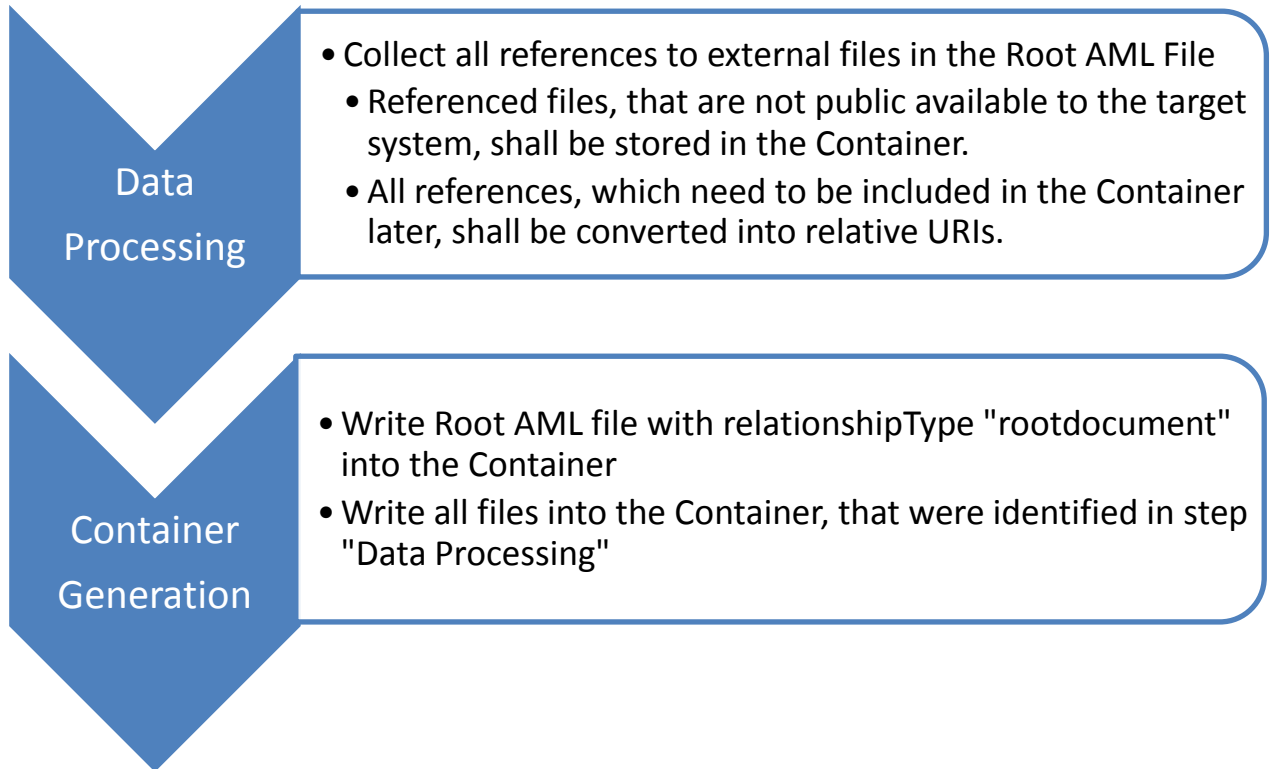


Figure 2 – Two Steps for the generation of an AutomationML Container

### 4 Example implementation

See AMLContainer.cs

## 5 Code examples

```
public void test()
{
    // create new container file
    using (AML.Container.AutomationMLContainer ac = new AML.Container.AutomationMLContainer("c:\\tmp\\testcontainer.xml"))
    {
        //add root file
        var root = ac.AddRoot("c:\\tmp\\test.xml", new Uri("/test.xml"));

        //add library file with relation to the root file
        ac.AddLibrary(root, "c:\\tmp\\Lib1.xml", new Uri("/Lib1.xml"));

        //enumerate root files
        foreach (var rootfile in ac.GetPartsByRelationType(AML.Container.RelationshipType.Root))
        {
            //enumerate library file related to the root file
            foreach (var rel in rootfile.GetRelationshipsByType(AML.Container.RelationshipType.Library.ToString))
            {
                //print relation
                Debug.Print("root: {0} --> library: {1}", rootfile.Uri.ToString, rel.TargetUri.ToString);
            }
        }
    }
}
```

Figure 3 – create new AML Container file

```
public void test()
{
    // open container file
    using (AML.Container.AutomationMLContainer ac = new AML.Container.AutomationMLContainer("c:\\tmp\\testcontainer.xml"))
    {
        //extract content
        ac.ExtractAllFiles("c:\\tmp\\");
    }
}
```

Figure 4 – extract an AML Container

## 6 References

/