IEC/PAS 61499-2

Edition 1.0
2001-05

PRE-STANDARD

Function blocks for industrial-process measurement and control systems –

Part 2: Software tools requirements

PUBLICLY AVAILABLE SPECIFICATION

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

Reference number
IEC/PAS 61499-2

IEC/PAS 61499-2

Edition 1.0
2001-05

PRE-STANDARD

Function blocks for industrial-process measurement and control systems –

Part 2: Software tools requirements

PUBLICLY AVAILABLE SPECIFICATION

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

Reference number
IEC/PAS 61499-2

TABLE OF CONTENTS

LIST OF TABLES

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FUNCTION BLOCKS FOR INDUSTRIAL-PROCESS MEASUREMENT AND CONTROL SYSTEMS

PART 2: SOFTWARE TOOLS REQUIREMENTS

FOREWORD

A PAS is a technical specification not fulfilling the requirements for a standard, but made available to the public and established in an organization operating under given procedures.

IEC-PAS 61499-2 has been processed by working group 6 of IEC technical committee 65: Industrial-process measurement and control.

The text of this PAS is based on the following document:

This PAS was approved for publication by the P-members of the committee concerned as indicated in the following document:

Draft PAS	Report on voting
65/260/PAS	65/265/RVD

Following publication of this PAS, the technical committee or subcommittee concerned will investigate the possibility of transforming the PAS into an International Standard.

This is Part 2 of a projected three-part Standard under development by Working Group 6 of IEC Technical Committee 65.

The projected parts of the standard are:

- Part 1 – Architecture.
- Part 2 – Software Tool Requirements.
- Part 3 – Application Guidelines

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.
3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
6) Attention is drawn to the possibility that some of the elements of this PAS may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

1. GENERAL REQUIREMENTS

1.1 Scope

This Standard consists of two Parts:

- Part 1, "Architecture", contains:
  - general requirements, including an introduction, scope, normative references, definitions, and reference models;
  - rules for the declaration of *function block types*, and rules for the behavior of *instances* of the types so declared;
  - rules for the use of function blocks in the *configuration* of distributed industrial-process measurement and control *systems* (IPMCSs);
  - rules for the use of function blocks in meeting the communication requirements of distributed IPMCSs;
  - rules for the use of function blocks in the management of *applications*, *resources* and *devices* in distributed IPMCSs;
  - requirements to be met by compliant systems and standards.
- This Part defines requirements for *software tools* to support the following systems engineering tasks enumerated in subclause 1.1 of IEC 61499-1:
  - the specification of *function block types*;
  - the functional specification of *resource types* and *device types*;
  - the specification, analysis, and validation of distributed IPMCSs;
  - the *configuration*, *implementation*, operation, and maintenance of distributed IPMCSs;
  - the exchange of *information* among *software tools*.

  It is assumed that such software tools may be used in the context of an Engineering Support System (ESS) as described in Annex C.1 of IEC 61499-1.
- Part 3, “Application Guidelines,” contains examples of the application of software tools in various stages of engineering methodologies for the performance of the tasks enumerated above.

It is beyond the scope of this Standard to specify the entire life cycle of industrial-process IPMCSs, or the entire set of tasks and activities required to support an IPCMS over its life cycle. However, other standards which do specify such tasks and activities may extend or modify the requirements specified in this Part.

1.2 Normative references

The normative references given in IEC 61499-1 also apply to this Part. In addition, the following document contains normative provisions that are used in an informative manner in Annex A of this Part:

17B/1022/CD, Draft IEC 61915: Low-voltage switchgear and controlgear - Profiles for networked industrial devices, 12 October 1999

All normative documents are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated above. Members of the IEC and ISO maintain registers of currently valid International Standards.

1.3 Definitions

The definitions given in IEC 61499-1 also apply to this Part. In addition, the following definitions apply for the purposes of this Part.

1.3.1. library element: The collection of *declarations* applying to a *data type, function block type, adapter type, subapplication type, resource type, device type*, or *system configuration*.

2. SOFTWARE TOOL REQUIREMENTS

This Clause defines the functional requirements of *software tools* which support the performance of the systems engineering tasks enumerated in subclause 1.1.

2.1 Information to be provided by the software tool supplier

The supplier of a *software tool* shall specify the following information in addition to other information which may be required in this Clause:

1. The type or types of *library element* to which the software tool applies.
2. The engineering task or tasks supported by the software tool. Task descriptions may be taken from the enumeration of engineering tasks given in subclause 1.1, or may be defined by the supplier.

2.2 Exchange of library elements

A *software tool* shall be capable of exchanging its *library elements* with other software tools. This exchange shall take the form of *data* in the format defined in Annex A, written on physical media or exchanged over communication links or networks

2.3 Information to be provided by the supplier of library elements

NOTE The provisions of this subclause are intended to provide the means by which the provider of a library element may achieve protection of intellectual property while still providing sufficient information to permit the effective use of the library element.

The provider of a *library element* may elect to provide an *implementation* of the library element.

EXAMPLE 1) The provider of a *function block type* library element may provide an implementation of the function block type as:

- one or more *instances* of the function block type in a *resource* contained in a *device* of Class 0 or higher as specified in IEC 61499-1-5.1;
- an instantiable implementation of the function block type in a *resource* contained in a *device* of Class 1 or higher as specified in IEC 61499-1-5.1;
- a file in an **implementation-dependent** format suitable for installation in a *resource* contained in a *device* of Class 2 as specified in IEC 61499-1-5.1, for instance using the IEC61499-FBMGT syntax defined in IEC 61499-1-F.3.1.2.

When an implementation of a library element is provided, the provider is not required to provide full details of the implementation. However, the provider shall provide sufficient information to enable the user to fully determine the functionality of the provided library element.

EXAMPLE 2) The requirement of the above paragraph would be met by the provider of an *instance* of a function block *type* in a *resource* through the provision, at a minimum, of the following information:

- a *function block type* library element specifying its *event* and *data interfaces* as defined in IEC 61499-1-2.2.1, and its *services* as defined in IEC 61499-1-3.1.2;
- *resource type* and *device type* library elements showing the occurrence and connections of the function block *instances*.

2.4 Display of declarations

A software tool shall be capable of displaying the *declarations* of its associated *library elements* in a form appropriate to the engineering task. This display may utilize the graphical or textual formats defined in IEC 61499-1, or a format defined by the supplier of the software tool.

NOTE The *declarations* of a library element may define its *interfaces* (event and data inputs and outputs) and internal *variables* as well as its *algorithms* and the control of their *execution*, for example via an *execution control chart* (ECC), etc.

2.5 Modification of declarations

A software tool shall enable its user to modify the declarations of its associated library elements as appropriate to the engineering task. Such modifications may include adding, deleting or changing the contents of declarations, and may be performed either graphically or textually or both.

2.6 Validation of declarations

If required by the associated engineering task, a software tool shall provide facilities for validation of the declarations of its associated library elements. Such facilities may include, but are not limited to:

1. Checking the correctness of the syntax of declarations.
2. Checking the semantic correctness of declarations, for instance, checking whether all *function block instances* in an *application* and its associated *sub applications* are properly allocated to *resources*, interconnected within resources, and intercommunicating among resources in a *system configuration.*
3. Simulation and testing of the operation of an *instance* of a library element *type*, either by itself or in association with other instances of the same or different types.

2.7 Implementation of declarations

If required by the associated engineering task, a software tool shall provide facilities for the *implementation* of the *declarations* of its associated *library elements*. Such facilities may include, but are not limited to:

1. The production of executable code ("firmware") for embedding in *instances* of *resource types* and *device types*.
2. The creation and interconnection ("downloading") of *function block instances* in *resources* and *devices*, for instance by using the management facilities defined in subclause 3.3 and Annexes F and G of IEC 61499-1.

2.8 System operation, testing and maintenance

If required by the associated engineering task, a software tool shall provide facilities for the operation, testing and maintenance of an Industrial Process Measurement and Control System (IPMCS) specified by its associated library elements. Such facilities may include, but are not limited to:

1. The facilities described in preceding subclauses of this Clause.
2. The information exchange facilities defined in subclause 3.2 and Annex F of IEC 61499-1.

ANNEX A - DOCUMENT TYPE DEFINITIONS (DTDs) (normative)

NOTE If there is a conflict between the provisions of this Annex and the provisions of Annex B of IEC 61499-1, the provisions of the latter shall prevail.

This Annex presents Document Type Descriptions (DTDs) for the exchange of IEC 61499 library elements between *software tools*. These DTDs are defined in the syntax defined in the eXtendedn Markup Language (XML) specification at **http://www.w3.org/TR/1998/REC-xml-19980210**.

The correspondences between the DTDs given in this Annex, the library elements defined in IEC 61499-1-C.1.1, and the textual syntax given in IEC 61499-1-B are given in Table A.0. The first table of each subclause of this Annex contains the DTD for the corresponding library element. The second table of each subclause provides a reference to the textual syntax (if any) plus an explanation for the major elements and attributes in the DTD. Following this, examples are given of the resulting XML files for typical library elements.

NOTE The examples given in this Annex provide a representative but not exhaustive sample of the features of the associated DTDs. In particular, these examples are not intended to be used as a test suite for compliance to the provisions of this Part.

Table A.0 - Document Type Definitions (DTDs)

DTD	LibraryElement	Textual Syntax
DataType	DataTypeDeclaration	`data_type_declaration` (IEC 61131-3-B.1.3)
FBType	FBTypeDeclaration	`fb_type_declaration`
SubapplicationType	SubapplicationTypeDeclaration	`subapplication_type_declaration`
AdapterType	AdapterTypeDeclaration	`adapter_type_declaration`
ResourceType	ResourceTypeDeclaration	`resource_type_specification`
DeviceType	DeviceTypeDeclaration	`device_type_specification`
System	SystemConfiguration	`system_configuration`

A.1 DataType

An XML document complying with the DTD in Table A.1-1 represents a **DataTypeDeclaration** object as described in Annex C.1.1 of IEC 61499-1.

Table A.1-1 - DataType DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT DataType
(Identification?,VersionInfo+,CompilerInfo?,ASN1Tag?,(DirectlyDerivedType
|EnumeratedType|SubrangeType|ArrayType|StructuredType))>
<!ATTLIST DataType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT VersionInfo EMPTY>
<!ATTLIST VersionInfo
 Organization CDATA #REQUIRED
 Version CDATA #REQUIRED
 Author CDATA #REQUIRED
 Date CDATA #REQUIRED
 Remarks CDATA #IMPLIED
>
<!ELEMENT ASN1Tag EMPTY>
<!ATTLIST ASN1Tag
 Class (UNIVERSAL | APPLICATION | CONTEXT | PRIVATE) #IMPLIED
 Number CDATA #REQUIRED
>
<!ELEMENT CompilerInfo (Compiler*)*>
<!ATTLIST CompilerInfo
 header CDATA #IMPLIED
 classdef CDATA #IMPLIED
>
<!ELEMENT Compiler EMPTY>
<!ATTLIST Compiler
 Language (Java | Cpp | C | Other) #REQUIRED
 Vendor CDATA #REQUIRED
 Product CDATA #REQUIRED
 Version CDATA #REQUIRED
>
<!ELEMENT DirectlyDerivedType EMPTY>
<!ATTLIST DirectlyDerivedType
 BaseType (BOOL|SINT|INT|DINT|LINT|USINT|UINT|UDINT|ULINT|REAL|LREAL|TIME
|DATE |TIME_OF_DAY|DATE_AND_TIME|STRING|BYTE|WORD|DWORD|LWORD|WSTRING)
#REQUIRED
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED
>
<!ELEMENT EnumeratedType (EnumeratedValue)+>
<!ATTLIST EnumeratedType
 InitialValue IDREF #IMPLIED
>
<!ELEMENT EnumeratedValue EMPTY>
<!ATTLIST EnumeratedValue
 Name ID #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT SubrangeType (Subrange)>
<!ATTLIST SubrangeType
 BaseType (SINT|INT|DINT|LINT|USINT|UINT|UDINT|ULINT) #REQUIRED
 InitialValue CDATA #IMPLIED
>
<!ELEMENT Subrange EMPTY>
<!ATTLIST Subrange
 LowerLimit CDATA #REQUIRED
 UpperLimit CDATA #REQUIRED
>
```

Table A.1-1 - DataType DTD

```
<!ELEMENT ArrayType (Subrange)+>
<!ATTLIST ArrayType
 BaseType CDATA #REQUIRED
 InitialValues CDATA #IMPLIED
 Comment CDATA #IMPLIED
>
<!ELEMENT StructuredType
(VarDeclaration|ArrayVarDeclaration|SubrangeVarDeclaration)+>
<!ATTLIST StructuredType
 Comment CDATA #IMPLIED
>
<!ELEMENT VarDeclaration EMPTY >
<!ATTLIST VarDeclaration
 Name ID #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
 InitialValue CDATA #IMPLIED
>
<!ELEMENT ArrayVarDeclaration (Subrange+) >
<!ATTLIST ArrayVarDeclaration
 Name ID #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
 InitialValues CDATA #IMPLIED
>
<!ELEMENT SubrangeVarDeclaration (Subrange?) >
<!ATTLIST SubrangeVarDeclaration
 Name ID #REQUIRED
 Type (SINT|INT|DINT|LINT|USINT|UINT|UDINT|ULINT) #REQUIRED
 Comment CDATA #IMPLIED
 InitialValue CDATA #IMPLIED
>
```

Explanations of the elements of the above DTD, and (where applicable) references to the formal syntax for their attributes, are given in Table A.1-2.

Table A.1-2 - DataType DTD Elements

Element Attributes	Textual Syntax (IEC 61131-3, Annex B)	Explanation
DataType		See IEC 61131-3-
Name	data_type_name	
Comment	--	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters
Identification	--	Information for data base retrieval
Standard	--	Primary reference standard in number-part-subclause format
Classification	--	Classification code as defined in reference standard
ApplicationDomain	--	Application domain as defined in reference standard
Function	--	Function of this element as defined in reference standard
Type	--	Element type (e.g., device type) as defined in reference standard
Description	--	Descriptive phrase as defined in reference standard

Table A.1-2 - DataType DTD Elements

Element Attributes	Textual Syntax (IEC 61131-3, Annex B)	Explanation
VersionInfo	--	Possibly one of several entries: • First entry - Most recent version • 2nd entry - Immediately preceding released version... • last entry - First released version
Organization	--	The organization supplying this library element
Version	digit [digit] '.' digit [digit] [letter]	The Version identification for this library element
Author	--	The author of this library element
Date	--	The release date of this version in YYYY-MM-DD format
Remarks	--	Comments relating to this version
ASN1Tag		ASN.1 tag per ISO/IEC 8824-5.8
Class		ASN.1 tag class per ISO/IEC 8824-5.8
Number		ASN.1 tag number per ISO/IEC 8824-5.8
CompilerInfo	--	Information for and about compilers used with this class
header	--	Header information such as package, imports, etc.
classdef	--	The class definition information such as superclass and implemented interfaces. If none is given, a default abstract superclass is used.
Compiler	--	Possibly one of several compilers used with this version
Language	--	The source language of this compiler
Vendor	--	The vendor of this compiler
Product	--	The product name of this compiler
Version	--	The version of this compiler
DirectlyDerivedType	See IEC 61131-3 Tables 12 and 14, #1	
BaseType	elementary_type_name	
InitialValue	constant	
EnumeratedType	See IEC 61131-3 Tables 12 and 14, #2	
InitialValue	enumerated_value	
EnumeratedValue	See IEC 61131-3 Table 14, #2	
Name	enumerated_value	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
SubrangeType	--	See IEC 61131-3 Tables 12 and 14, #3
BaseType	integer_type_name	
InitialValue	signed_integer	
Subrange	See IEC 61131-3 Tables 12 and 14, #3	
LowerLimit	signed_integer	
UpperLimit	signed_integer	

Table A.1-2 - DataType DTD Elements

Element Attributes	Textual Syntax (IEC 61131-3, Annex B)	Explanation
`ArrayType`	See IEC 61131-3 Tables 12 and 14, #4	
`BaseType`	`non_generic_type_name`	
`InitialValues`	`array_initialization`	
`StructuredType`	See IEC 61131-3 Tables 12, #5 and 14, #5 and #6	
`VarDeclaration`	See NOTE	
`Name`	`structure_element_name`	
`Type`	`elementary_type_name | simple_type_name | enumerated_type_name | structure_type_name`	See NOTE
`Comment`	A comment per IEC 61131-3-2.1.5 without `(*` and `*)` delimiters	
`InitialValue`	`constant | enumerated_value | initialized_structure`	See NOTE
`ArrayVarDeclaration`	See IEC 61131-3-2.3.3.	
`Name`	`structure_element_name`	
`Type`	`array_type_name`	
`Comment`	A comment per IEC 61131-3-2.1.5 without `(*` and `*)` delimiters	
`InitialValues`	`array_initialization`	
`SubrangeVarDeclaration`	See IEC 61131-3-2.3.3.	
`Name`	`structure_element_name`	
`Type`	`integer_type_name`	
`Comment`	A comment per IEC 61131-3-2.1.5 without `(*` and `*)` delimiters	
`InitialValue`	`signed_integer`	

NOTE - The `Type` and `InitialValue` syntax for this element must match each other according to the IEC 61131-3 Annex B syntax for `structure_element_declaration`.

EXAMPLE - The structured data type `ANALOG_CHANNEL_CONFIGURATIONI` example is expressed in IEC 61131-3, Table 14 as follows:

```
TYPE ANALOG_CHANNEL_CONFIGURATIONI :
    STRUCT
      RANGE : ANALOG_SIGNAL_RANGE ;
      MIN_SCALE : ANALOG_DATA := -4095 ;
      MAX_SCALE : ANALOG_DATA :=  4095 ;
    END_STRUCT ;
END_TYPE
```

A corresponding XML document could be:


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataType SYSTEM "DataType.dtd" >
<DataType
   Name="ANALOG_CHANNEL_CONFIGURATIONI"
   Comment="IEC 61131-3, Table 14#5">
<Identification
   Function="Configuration Data"
   Standard="61131-3-2.3.3.2"
   ApplicationDomain="Any"
   Classification="Data type"
   Type="Analog"
   Description="Table 14, #5"/>
<VersionInfo
   Organization="IEC SC65B/WG7/TF3"
   Version="2.0"
   Author="JHC"
   Date="2000-01-31"/>
<StructuredType>
  <VarDeclaration Name="SIGNAL_RANGE"
     Type="ANALOG_SIGNAL_RANGE"/>
  <VarDeclaration Name="MIN_SCALE"
     Type="ANALOG_DATA"
     InitialValue="-4095"/>
  <VarDeclaration Name="MAX_SCALE"
     Type="ANALOG_DATA"
     InitialValue="4095"/>
</StructuredType>
</DataType>
```

A.2 LibraryElement

An XML document complying with the DTD in Table A.2-1 represents a **LibraryElement** object as described in Annex C.1.1 of IEC 61499-1. Possible root elements of such a document are `FBType`, `AdapterType`, `ResourceType`, `DeviceType`, `System`, and `SubappType`, representing the concrete subclasses **FBTypeDeclaration, AdapterTypeDeclaration, ResourceTypeDeclaration, DeviceTypeDeclaration, SystemConfiguration, and SubapplicationTypeDeclaration** of the abstract superclass **LibraryElement**, respectively. The **DataTypeDeclaration** subclass is represented separately by the DTD given in Annex A.1 of this specification.

Table A.2-1 - Library Element DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Common elements -->
<!ELEMENT Identification EMPTY>
<!ATTLIST Identification
 Standard CDATA #IMPLIED
 Classification CDATA #IMPLIED
 ApplicationDomain CDATA #IMPLIED
 Function CDATA #IMPLIED
 Type CDATA #IMPLIED
 Description CDATA #IMPLIED
>
```

```
<!ELEMENT VersionInfo EMPTY>
<!ATTLIST VersionInfo
 Organization CDATA #REQUIRED
 Version CDATA #REQUIRED
 Author CDATA #REQUIRED
 Date CDATA #REQUIRED
 Remarks CDATA #IMPLIED
>
```

```
<!ELEMENT CompilerInfo (Compiler*)>
<!ATTLIST CompilerInfo
 header CDATA #IMPLIED
 classdef CDATA #IMPLIED
>
<!ELEMENT Compiler EMPTY>
<!ATTLIST Compiler
 Language (Java | Cpp | C | Other) #REQUIRED
 Vendor CDATA #REQUIRED
 Product CDATA #REQUIRED
 Version CDATA #REQUIRED
>
```

```
<!ELEMENT FBNetwork (FB*,EventConnections?,DataConnections?,AdapterConnections?)>
<!ELEMENT FB EMPTY>
<!ATTLIST FB
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
  x CDATA #IMPLIED
  y CDATA #IMPLIED
>
```

```
<!ELEMENT EventConnections (Connection+)>
<!ELEMENT DataConnections (Connection+)>
<!ELEMENT AdapterConnections (Connection+)>
<!ELEMENT Connection EMPTY>
<!ATTLIST Connection
 Source CDATA #REQUIRED
 Destination CDATA #REQUIRED
 Comment CDATA #IMPLIED
 dx1 CDATA #IMPLIED
 dx2 CDATA #IMPLIED
 dy CDATA #IMPLIED
>
```

Table A.2-1 - Library Element DTD

```
<!-- FBType elements -->
<!ELEMENT FBType (Identification?,VersionInfo+,CompilerInfo?,InterfaceList,
 (BasicFB | FBNetwork)?, Service?) >
<!ATTLIST FBType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
```

```
<!ELEMENT InterfaceList (EventInputs?,EventOutputs?,InputVars?,OutputVars?,
 Sockets?, Plugs?)>
<!ELEMENT EventInputs (Event+)>
<!ELEMENT EventOutputs (Event+)>
<!ELEMENT InputVars (VarDeclaration+)>
<!ELEMENT OutputVars (VarDeclaration+)>
<!ELEMENT Sockets (AdapterDeclaration+)>
<!ELEMENT Plugs (AdapterDeclaration+)>
```

```
<!ELEMENT Event (With*)>
<!ATTLIST Event
 Name ID #REQUIRED
 Type CDATA #IMPLIED
 Comment CDATA #IMPLIED
>
<!ELEMENT With EMPTY>
<!ATTLIST With
 Var IDREF #REQUIRED
>
```

```
<!ELEMENT VarDeclaration EMPTY>
<!ATTLIST VarDeclaration
 Name ID #REQUIRED
 Type CDATA #REQUIRED
 ArraySize CDATA #IMPLIED
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED
>
```

```
<!ELEMENT AdapterDeclaration EMPTY>
<!ATTLIST AdapterDeclaration
 Name ID #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
```

```
<!ELEMENT BasicFB (InternalVars?,ECC?,Algorithm*)>
<!ELEMENT InternalVars (VarDeclaration+)>
<!ELEMENT ECC (ECState+ ECTransition+)
>
```

```
<!ELEMENT ECState (ECAction*)>
<!ATTLIST ECState
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED
>
```

```
<!ELEMENT ECTransition EMPTY>
<!ATTLIST ECTransition
 Source CDATA #REQUIRED
 Destination CDATA #REQUIRED
 Condition CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED
>
```

```
<!ELEMENT ECAction EMPTY>
<!ATTLIST ECAction
 Algorithm CDATA #IMPLIED
 Output IDREF #IMPLIED
>
```

Table A.2-1 - Library Element DTD

```
<!ELEMENT Algorithm ((FBD | ST | LD | Other))>
<!ATTLIST Algorithm
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT FBD (FB+,DataConnections)
>
<!ELEMENT ST EMPTY>
<!ATTLIST ST
 Text CDATA #REQUIRED
>
<!ELEMENT LD (Rung+)
>
<!ELEMENT Rung EMPTY>
<!ATTLIST Rung
 Output IDREF #REQUIRED
 Expression CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT Other EMPTY>
<!ATTLIST Other
 Language CDATA #REQUIRED
 Text CDATA #REQUIRED
>
<!ELEMENT Service (ServiceSequence+)>
<!ATTLIST Service
 RightInterface CDATA #REQUIRED
 LeftInterface CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT ServiceSequence (ServiceTransaction*)>
<!ATTLIST ServiceSequence
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT ServiceTransaction (InputPrimitive?, OutputPrimitive*)>

<!ELEMENT InputPrimitive EMPTY>
<!ATTLIST InputPrimitive
 Interface CDATA #REQUIRED
 Event CDATA #REQUIRED
 Parameters CDATA #IMPLIED
>
<!ELEMENT OutputPrimitive EMPTY>
<!ATTLIST OutputPrimitive
 Interface CDATA #REQUIRED
 Event CDATA #REQUIRED
 Parameters CDATA #IMPLIED
>
<!-- AdapterType elements -->
<!ELEMENT AdapterType
(Identification?,VersionInfo+,CompilerInfo?,InterfaceList,Service?)>
<!ATTLIST AdapterType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!-- ResourceType elements -->
<!ELEMENT ResourceType (Identification?,VersionInfo+, CompilerInfo?, FBTypeName*,
VarDeclaration*, FBNetwork)>
<!ATTLIST ResourceType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT FBTypeName EMPTY>
<!ATTLIST FBTypeName
 Name CDATA #REQUIRED
>
```

Table A.2-1 - Library Element DTD

```
<!-- DeviceType elements -->
<!ELEMENT DeviceType (Identification?,VersionInfo+, CompilerInfo?, VarDeclaration*,
ResourceTypeName*, Resource*, DeviceFBNetwork?)>
<!ATTLIST DeviceType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT ResourceTypeName EMPTY>
<!ATTLIST ResourceTypeName
 Name CDATA #REQUIRED
>
<!ELEMENT Resource (FBNetwork?)>
<!ATTLIST Resource
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED
>
```

```
<!-- System elements -->
<!ELEMENT System
(Identification?,VersionInfo+,Application*,Device+,DeviceParameters?,Mapping*)>
<!ATTLIST System
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT Application (FBNetwork)>
<!ATTLIST Application
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT Mapping EMPTY>
<!ATTLIST Mapping
 From CDATA #REQUIRED
 To CDATA #REQUIRED
>
<!ELEMENT Device (Resource*,FBNetwork?)>
<!ATTLIST Device
 Name ID #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED
>
<!ELEMENT DeviceParameters (Connection+)>
```

Table A.2-1 - Library Element DTD

```
<!-- SubAppType elements -->
<!ELEMENT SubAppType (Identification?,VersionInfo+,CompilerInfo?,SubAppInterfaceList,
SubAppNetwork?)>
<!ATTLIST SubAppType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
<!ELEMENT SubAppInterfaceList
(SubAppEventInputs?,SubAppEventOutputs?,InputVars?,OutputVars?)>
<!ELEMENT SubAppEventInputs (SubAppEvent+)>
<!ELEMENT SubAppEventOutputs (SubAppEvent+)>
<!ELEMENT SubAppEvent EMPTY>
<!ATTLIST SubAppEvent
 Name ID #REQUIRED
 Type CDATA #IMPLIED
 Comment CDATA #IMPLIED
>
<!ELEMENT SubAppNetwork (SubApp*,FB*,EventConnections?,DataConnections?)>
<!ELEMENT SubApp EMPTY>
<!ATTLIST SubApp
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
  x CDATA #IMPLIED
  y CDATA #IMPLIED
>
```

Explanations of some of the elements of the above DTD, and (where applicable) references to the formal syntax for their attributes, are given in Table A.2-2.

Table A.2-2 - LibraryElement DTD Elements

Element Attributes	Syntax (IEC 61499-1, Annex B.2)	Explanation
Identification	See Table A.1-2	
VersionInfo		
CompilerInfo		
Compiler		
FBNetwork	A *function block network* as defined in IEC 61499-1.	
FB	A *function block instance* as defined in IEC 61499-1.	
Name	fb_instance_reference	See NOTE 3.
Type	fb_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
x, y	See Annex B.	
Connection	An *event connection, data connection* or *adapter connection* as defined in IEC 61499-1.	
Source	See NOTES 3,4.	
Destination	See NOTES 3,4.	
dx1, dx2, dy	See Annex B.	
FBType	An **FBTypeDeclaration** as described in IEC 61499-1-C.1.1.	
Name	fb_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	

Table A.2-2 - LibraryElement DTD Elements

Element Attributes	Syntax (IEC 61499-1, Annex B.2)	Explanation
Event	A *declaration* of an *event interface.*	
Name	event_input_name | event_output_name	See NOTE 5.
Type	event_type	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
With	A *declaration* of an association between an *event* and a *variable.*	
Var	input_variable_name | output_ variable _name	See NOTE 5.
VarDeclaration	A *declaration* of a *variable.*	
Name	input_variable_name | output_variable_name | internal_variable_name	See NOTE 6.
Type	identifier	
ArraySize	See NOTE 7.	
InitialValue	See NOTE 8.	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
AdapterDeclaration	A *declaration* of a *plug* or *socket interface* of a *function block type.*	
Name	plug_name | socket_name	See NOTE 9.
Type	adapter_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
ECState	An *EC state* as defined in IEC 61499-1.	
Name	ec_state_name	See NOTE 3.
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
x, y	See Annex B.	
ECTransition	An *EC transition* as defined in IEC 61499-1.	
Source	ec_state_name	
Destination	ec_state_name	
Condition	ec_transition_condition	
x, y	See Annex B.	
ECAction	An *EC action* as defined in IEC 61499-1.	
Algorithm	algorithm_name	
Output	event_output_name	
Algorithm	An *algorithm* in a specified language (NOTE 1)	
Name	algorithm_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
ST	An *algorithm* in the IEC 61131-3 ST language	
Text	statement_list	per IEC 61131-3-B.3.2.

Table A.2-2 - LibraryElement DTD Elements

Element Attributes	Syntax (IEC 61499-1, Annex B.2)	Explanation
Rung	A rung of an algorithm in the LD language	
Output	See NOTE 2	
Expression	See NOTE 2	
Other	An algorithm in a language other than FBD, ST or LD.	
Language	The name of the programming language; see NOTE 1	
Text	The algorithm text in the programming language; see NOTE 1	
Service	A *declaration* of a *service* per IEC 61499-1-3.1.2	
RightInterface	service_interface_name	
LeftInterface	service_interface_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
ServiceSequence	A *declaration* of a *service sequence* per IEC 61499-1-3.1.2	
Name	sequence_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
InputPrimitive	An "input" *service primitive* per IEC 61499-1-3.1.2	
Interface	service_interface_name	
Event	(([plug_name '.'] event_input_name) | (socket_name '.' event_output_name)) ['+' | '-']	
Parameters	input_variable_name {',' input_variable_name}	
OutputPrimitive	An "output" *service primitive* per IEC 61499-1-3.1.2	
Interface	service_interface_name	
Event	('NULL' | ([plug_name '.'] event_output_name) | socket_name '.' event_input_name)) ['+' | '-']	
Parameters	output_variable_name {',' output_variable_name}	
AdapterType	A *declaration* of an *adapter interface type* per IEC 61499-1-2.5	
Name	adapter_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
ResourceType	A *declaration* of a *resource type* per IEC 61499-1-4.1.1	
Name	resource_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
FBTypeName	The name of a *function block type* supported by all *instances* of a *resource type*	
Name	fb_type_name	
DeviceType	A *declaration* of a *device type* per IEC 61499-1-4.1.2	
Name	device_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	

Table A.2-2 - LibraryElement DTD Elements

Element Attributes	Syntax (IEC 61499-1, Annex B.2)	Explanation
ResourceTypeName	The name of a *resource type* supported by all *instances* of a *device type*	
Name	resource_type_name	
Resource	*A resource instance* present in all *instances* of a *device type*	
Name	resource_instance_name	
Type	resource _type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
System	A *declaration* of a *system configuration* per IEC 61499-1-4.2	
Name	system_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
Application	A *declaration* of an *application* per IEC 61499-1-4.2.2	
Name	application_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
Mapping	Mapping of a *function block instance* from an *application* onto a function block instance in a *resource*.	
From	fb_instance_reference	Hierarchical *function block instance name* in its *application*, e.g., APP1.SUBAPP2.FB2
To	fb_resource_reference	Hierarchical *function block instance name* in the physical system, e.g., DEV1.RES2.FB2
Device	A *declaration* of a *device configuration* per IEC 61499-1-4.2.3	
Name	device_instance_name	
Type	device_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
DeviceParameters	*Declarations* of the values of *parameters* of a *device instance*.	
SubAppType	A *declaration* of a *subapplication type* per IEC 61499-1-2.4	
Name	subapp_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
SubAppEvent	A *declaration* of an *event interface* of a *subapplication type.*	
Name	event_input_name | event_output_name	See NOTE 10.
Type	event_type	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
SubApp	A *subapplication instance* as defined in IEC 61499-1.	
Name	subapp_instance_name	See NOTE 3.
Type	subapp_type_name	
Comment	A comment per IEC 61131-3-2.1.5 without (* and *) delimiters	
x, y	See Annex B.	

Table A.2-2 - LibraryElement DTD Elements

Element Attributes	Syntax (IEC 61499-1, Annex B.2)	Explanation

NOTE 1 - The specification of algorithms in languages other than FBD, ST and LD is beyond the scope of this Part of IEC 61499.

NOTE 2 - Since the FBD and ST languages are available for the specification of complex algorithms, it is recommended that the usage of the LD language in the context of this Standard be limited to rungs performing the evaluation of assignment statements of the form `<output> := <expression>`. For portability between software tools, it is further recommended that the XML `Expression` element have the following simple postfix-operator textual syntax with whitespace_separated terms:

```
expression ::= and_expression
and_expression ::= (variable_name ['!']) | or_expression
      | and_expression and_expression '&'
or_expression ::= and_expression | or_expression or_expression '|'
```

See EXAMPLE 1 for an illustration of this recommended usage.

NOTE 3 - The appropriate subset of the production `fb_instance_reference` should be used, depending on the context of the `FBNetwork` element. For instance, in the context of an `FBType` element, only the `fb_instance_name` portion of the reference would be used.

NOTE 4 – Depending on the context, the syntax of a `Source` or `Destination` element should correspond to the syntax of the respective element in one of the productions `event_conn, data_conn, adapter_conn, subapp_event_conn, subapp_data_conn, config_event_conn, config_data_conn, config_adapter_conn, devtype_event_conn, devtype_data_conn,` or `devtype_adapter_conn` given in Annex B of IEC 61499-1.

NOTE 5 – The productions `event_input_name` and `input_variable_name` apply when the `Event` element is part of an `EventInputs` element, and `event_output_name` and `output_variable_name` apply when it is part of an `EventOutputs` element.

NOTE 6 - The productions `input_variable_name, output_variable_name` and `internal_variable_name` apply when the associated `VarDeclaration` element is part of an `InputVars, OutputVars` or `InternalVars` element, respectively.

NOTE 7- The syntax of this element when present shall be equivalent to the syntactic expression `(subrange {',' subrange}) | integer {',' integer}` where the non-terminals `subrange` and `integer` are as defined in Annex B of IEC 61131-3. Each term of the second form is equivalent to the subrange `0..n-1`, where n is the value of the corresponding `integer` syntactic element. If this element is missing, the variable is not an array.

NOTE 8 - The syntax of this element is the syntax for initialization of the corresponding variable type as defined in Annex B.1.4.3 of IEC 61131-3.

NOTE 9 - The productions `plug_name` and `socket_name` apply when the associated `AdapterDeclaration` element is part of a `Plugs` or `Sockets` element, respectively.

NOTE 10 - The productions `event_input_name` and `event_output_name` apply when the `SubAppEvent` element is part of a `SubAppEventInputs` or `SubAppEventOutputs` element, respectively.

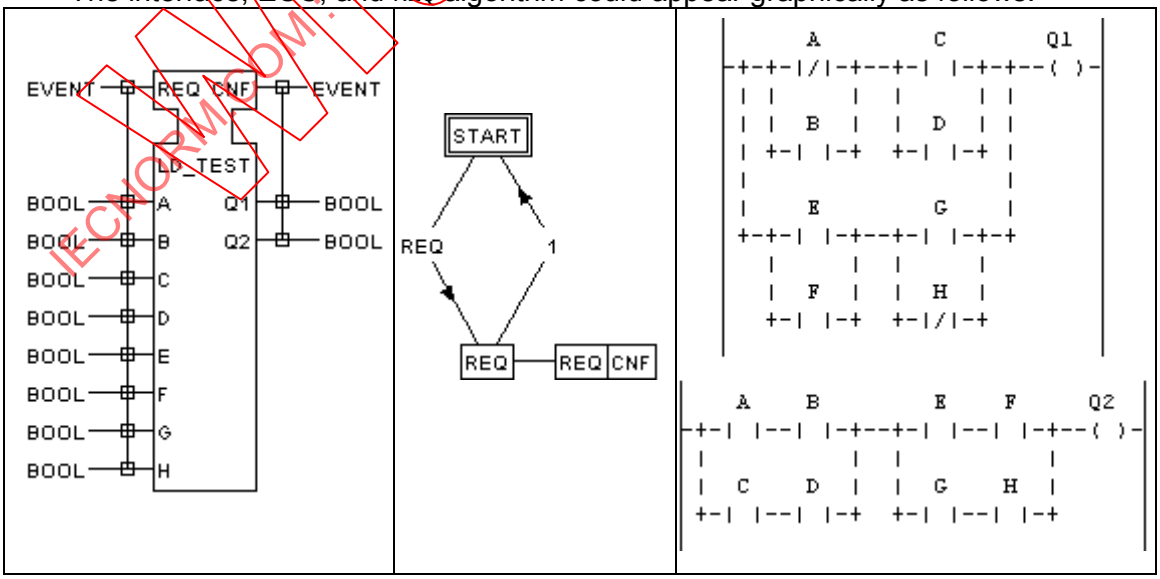
A.3 Examples

A.3.1 Basic function block types

EXAMPLE 1. A basic function block type containing an LD algorithm according to NOTE 2 of Table A.2-2 could be expressed textually as follows:

```
FUNCTION_BLOCK LD_TEST (* LD Algorithm Example *)
EVENT_INPUT
    REQ WITH A, B, C, D, E, F, G, H;
END_EVENT
EVENT_OUTPUT
    CNF WITH Q1, Q2; (* Execution Confirmation *)
END_EVENT
VAR_INPUT
    A : BOOL;
    B : BOOL;
    C : BOOL;
    D : BOOL;
    E : BOOL;
    F : BOOL;
    G : BOOL;
    H : BOOL;
END_VAR
VAR_OUTPUT
    Q1 : BOOL;
    Q2 : BOOL;
END_VAR
EC_STATES
    START ; (* Initial State *)
    REQ : REQ -> CNF ; (* Normal execution *)
END_STATES
EC_TRANSITIONS
    START TO REQ := REQ;
    REQ TO START := 1;
END_TRANSITIONS
ALGORITHM REQ IN LD :
    Q1 := ((!A|B)&(C|D))|((E|F)&(G|!H));
    Q2 := ((A&B)|(C&D))&((E&F)|(G&H));
END_ALGORITHM
END_FUNCTION_BLOCK
```

The interface, ECC, and REQ algorithm could appear graphically as follows:



A corresponding XML document would be:

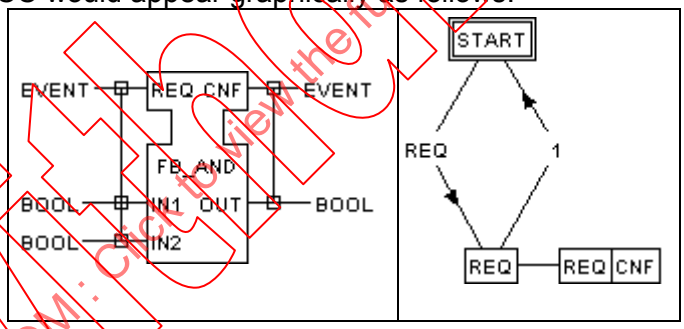
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="LD_TEST" Comment="LD Algorithm Example" >
  <Identification Standard="61499-2-A.3.1" Description="LD Algorithm Example"
```

```
/>
  <VersionInfo Organization="IEC TC65/WG6" Version="0.2" Author="JHC"
Date="2000-11-16" Remarks="Corrected Identification" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-06-20" Remarks="Tested Sun compiler" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-02-01" />
  <CompilerInfo header="package fb.rt.part2;" >
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge" Version="3.0"
/>
    <Compiler Language="Java" Vendor="Sun" Product="JDK" Version="1.1.8" />
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="REQ" >
        <With Var="A" />
        <With Var="B" />
        <With Var="C" />
        <With Var="D" />
        <With Var="E" />
        <With Var="F" />
        <With Var="G" />
        <With Var="H" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" Comment="Execution Confirmation" >
        <With Var="Q1" />
        <With Var="Q2" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="A" Type="BOOL" />
      <VarDeclaration Name="B" Type="BOOL" />
      <VarDeclaration Name="C" Type="BOOL" />
      <VarDeclaration Name="D" Type="BOOL" />
      <VarDeclaration Name="E" Type="BOOL" />
      <VarDeclaration Name="F" Type="BOOL" />
      <VarDeclaration Name="G" Type="BOOL" />
      <VarDeclaration Name="H" Type="BOOL" />
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="Q1" Type="BOOL" />
      <VarDeclaration Name="Q2" Type="BOOL" />
    </OutputVars>
  </InterfaceList>
  <BasicFB>
    <ECC >
      <ECState Name="START" Comment="Initial State" x="341.1765" y="105.8824"
>
      </ECState>
      <ECState Name="REQ" Comment="Normal execution" x="358.8235"
y="858.8235" >
        <ECAction Algorithm="REQ" Output="CNF" />
      </ECState>
      <ECTransition Source="START" Destination="REQ" Condition="REQ"
x="170.5882" y="494.1176" />
      <ECTransition Source="REQ" Destination="START" Condition="1"
x="564.7059" y="500" />
    </ECC>
  <Algorithm Name="REQ" Comment="Normally executed algorithm" >
    <LD >
      <Rung Output="Q1" Expression="A ! B | C D | &#38; E F | G H ! | &#38; |
" />
      <Rung Output="Q2" Expression="A B &#38; C D &#38; | E F &#38; G H &#38;
| &#38; " />
    </LD>
  </Algorithm>
  </BasicFB>
</FBType>
```

EXAMPLE 2. A basic function block type containing a ST algorithm could be expressed textually as follows:

```
FUNCTION_BLOCK FB_AND (* Boolean AND *)
EVENT_INPUT
    REQ WITH IN1, IN2;
END_EVENT
EVENT_OUTPUT
    CNF WITH OUT;
END_EVENT
VAR_INPUT
    IN1 : BOOL;
    IN2 : BOOL;
END_VAR
VAR_OUTPUT
    OUT : BOOL; (* IN1&IN2 *)
END_VAR
EC_STATES
    START ; (* Initial State *)
    REQ : REQ -> CNF ; (* Normal execution
*)
END_STATES
EC_TRANSITIONS
    START TO REQ := REQ;
    REQ TO START := 1;
END_TRANSITIONS
ALGORITHM REQ IN ST :
  OUT := (IN1 & IN2);
END_ALGORITHM
END_FUNCTION_BLOCK
```

The interface and ECC would appear graphically as follows:



A corresponding XML document would be:

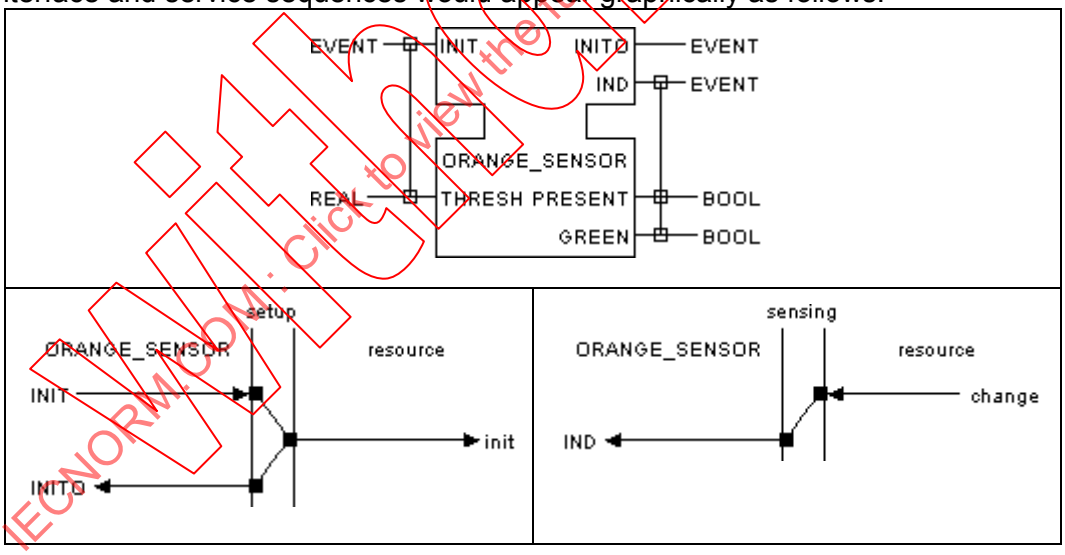
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="FB_AND" Comment="Boolean AND" >
  <Identification Standard="61499-1-D.1" Classification="Math"
ApplicationDomain="Any" Function="AND" Type="Boolean" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1"
Author="JHC" Date="2000-06-10" Remarks="Tested Sun compiler." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0"
Author="JHC" Date="2000-01-29" Remarks="Simple Boolean AND" />
  <CompilerInfo header="package fb.rt.part2;" >
    <Compiler Language="Java" Vendor="Sun" Product="JDK"
Version="1.1.8" />
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge"
Version="3.0" />
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="REQ" >
        <With Var="IN1" />
        <With Var="IN2" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" >
        <With Var="OUT" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="IN1" Type="BOOL" />
      <VarDeclaration Name="IN2" Type="BOOL" />
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="OUT" Type="BOOL" Comment="IN1&#38;IN2"
/>
    </OutputVars>
  </InterfaceList>
  <BasicFB>
    <ECC >
      <ECState Name="START" Comment="Initial State" x="200"
y="105.8824" >
      </ECState>
      <ECState Name="REQ" Comment="Normal execution" x="205.8824"
y="676.4706" >
        <ECAction Algorithm="REQ" Output="CNF" />
      </ECState>
      <ECTransition Source="START" Destination="REQ"
Condition="REQ" x="370.5882" y="405.8824" />
      <ECTransition Source="REQ" Destination="START" Condition="1"
x="52.9412" y="429.4117" />
    </ECC>
  <Algorithm Name="REQ" >
    <ST Text="  OUT := (IN1 &#38; IN2);&#10;" />
  </Algorithm>
  </BasicFB>
</FBType>
```


A.3.2 Service interface function block types

EXAMPLE 1. A service interface function block type for sensing the presence and condition of an orange on a conveyor could be expressed textually as follows:

```
FUNCTION_BLOCK ORANGE_SENSOR (* Sense Presence & Color of Orange *)
EVENT_INPUT
    INIT WITH THRESH; (* Set Threshold *)
END_EVENT
EVENT_OUTPUT
    INITO; (* Threshold Set *)
    IND WITH PRESENT, GREEN; (* Change in Presence or Color *)
END_EVENT
VAR_INPUT
    THRESH : REAL; (* Adjustable Color Threshold *)
END_VAR
VAR_OUTPUT
    PRESENT : BOOL; (* Orange is Present *)
    GREEN : BOOL; (* Green is Above Threshold *)
END_VAR
SERVICE ORANGE_SENSOR/resource
SEQUENCE setup
    ORANGE_SENSOR.INIT(THRESH) -> resource.init() ->
ORANGE_SENSOR.INITO();
END_SEQUENCE
SEQUENCE sensing
    resource.change() -> ORANGE_SENSOR.IND(PRESENT,GREEN);
END_SEQUENCE
END_SERVICE
END_FUNCTION_BLOCK
```

The interface and service sequences would appear graphically as follows:



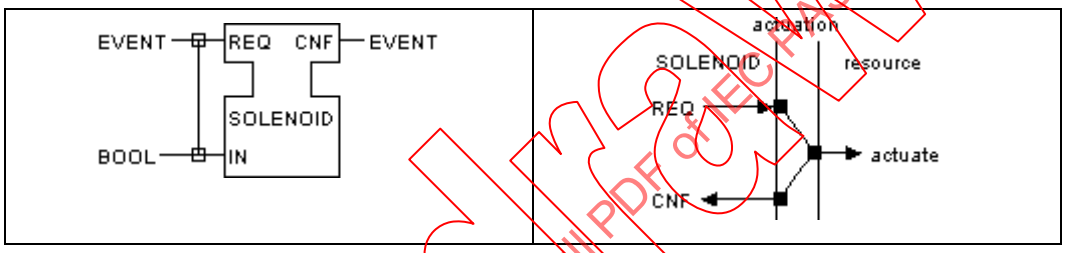
A corresponding XML document would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="ORANGE_SENSOR" Comment="Sense Presence &#38; Color of Orange"
>
  <Identification Standard="61915" Classification="C0202"
ApplicationDomain="Food Processing" Function="Detection"
Type="Photoelectric Sensors" Description="Orange Presence and Quality" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-05-14" Remarks="Modified to use LibraryElement.dtd" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-01-26" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="INIT" Comment="Set Threshold" >
        <With Var="THRESH" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="INITO" Comment="Threshold Set" >
      </Event>
      <Event Name="IND" Comment="Change in Presence or Color" >
        <With Var="PRESENT" />
        <With Var="GREEN" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="THRESH" Type="REAL" Comment="Adjustable Color
Threshold" />
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="PRESENT" Type="BOOL" Comment="Orange is
Present" />
      <VarDeclaration Name="GREEN" Type="BOOL" Comment="Green is Above
Threshold" />
    </OutputVars>
  </InterfaceList>
  <Service RightInterface="resource" LeftInterface="ORANGE_SENSOR" >
    <ServiceSequence Name="setup" >
      <ServiceTransaction >
        <InputPrimitive Interface="ORANGE_SENSOR" Event="INIT"
Parameters="THRESH" />
        <OutputPrimitive Interface="resource" Event="init" />
        <OutputPrimitive Interface="ORANGE_SENSOR" Event="INITO" />
      </ServiceTransaction>
    </ServiceSequence>
    <ServiceSequence Name="sensing" >
      <ServiceTransaction >
        <InputPrimitive Interface="resource" Event="change" />
        <OutputPrimitive Interface="ORANGE_SENSOR" Event="IND"
Parameters="PRESENT,GREEN" />
      </ServiceTransaction>
    </ServiceSequence>
  </Service>
</FBType>
```


EXAMPLE 2. A service interface function block type for the actuator of a simple solenoid valve could be expressed textually as follows:

```
FUNCTION_BLOCK SOLENOID (* Solenoid Valve *)
EVENT_INPUT
    REQ WITH IN; (* Set Actuator Status *)
END_EVENT
EVENT_OUTPUT
    CNF; (* Actuator Status Change Confirmed *)
END_EVENT
VAR_INPUT
    IN : BOOL; (* Actuator Value,1=OPEN,0=CLOSED *)
END_VAR
SERVICE SOLENOID/resource
SEQUENCE actuation
    SOLENOID.REQ(IN) -> resource.actuate(IN) -> SOLENOID.CNF();
END_SEQUENCE
END_SERVICE
END_FUNCTION_BLOCK
```

The interface and service sequence would appear graphically as follows:



A corresponding XML document would be:

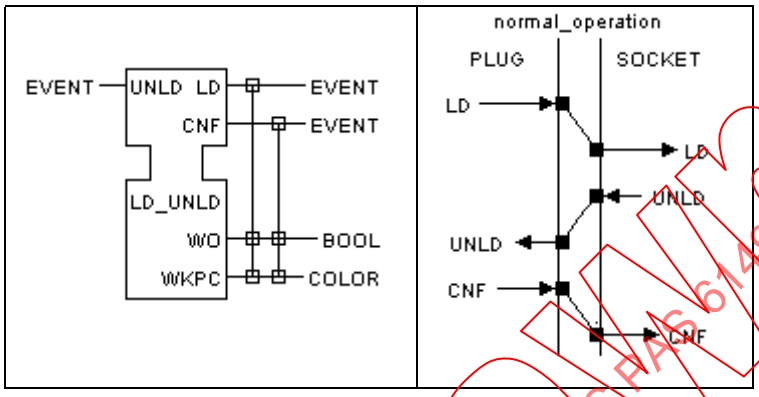
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="SOLENOID" Comment="Solenoid Valve" >
  <Identification Standard="61915" Classification="C0403"
ApplicationDomain="Any" Function="Logic / I/O Modules &#38; Controllers"
Type="Actuators" Description="Solenoid Valve" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-02-03" Remarks="Corrected service sequence" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-01-26" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="REQ" Comment="Set Actuator Status" >
        <With Var="IN" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" Comment="Actuator Status Change Confirmed" >
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="IN" Type="BOOL" Comment="Actuator
Value,1=OPEN,0=CLOSED" />
    </InputVars>
  </InterfaceList>
  <Service RightInterface="resource" LeftInterface="SOLENOID" >
    <ServiceSequence Name="actuation" >
      <ServiceTransaction >
        <InputPrimitive Interface="SOLENOID" Event="REQ" Parameters="IN"
/>
        <OutputPrimitive Interface="resource" Event="actuate"
Parameters="IN" />
```



```
          <OutputPrimitive Interface="SOLENOID" Event="CNF" />
        </ServiceTransaction>
      </ServiceSequence>
    </Service>
  </FBType>
```

A.3.3 An adapter interface type

EXAMPLE. An adapter interface for use in parts transfer simulations and its typical sequence of operation, and its corresponding XML document, may be as shown below.




```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AdapterType SYSTEM "../LibraryElement.dtd" >
<AdapterType Name="LD_UNLD" Comment="LOAD/UNLOAD Adapter Interface" >
  <Identification Standard="61499-2" />
  <VersionInfo Organization="Rockwell Automation" Version="0.0" Author="JHC"
Date="1999-11-17" Remarks="Generated by FBEditor application" />
  <CompilerInfo header="package fb.rt.omac;" >
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge" Version="2.0"
/>
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="UNLD" Comment="UNLOAD Request" >
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="LD" Comment="LOAD Request" >
        <With Var="WO" />
        <With Var="WKPC" />
      </Event>
      <Event Name="CNF" Comment="UNLD Confirm" >
        <With Var="WO" />
        <With Var="WKPC" />
      </Event>
    </EventOutputs>
    <OutputVars>
      <VarDeclaration Name="WO" Type="BOOL" Comment="Workpiece present" />
      <VarDeclaration Name="WKPC" Type="COLOR" Comment="Workpiece Color" />
    </OutputVars>
  </InterfaceList>
  <Service RightInterface="SOCKET" LeftInterface="PLUG" >
    <ServiceSequence Name="normal_operation" >
      <ServiceTransaction >
        <InputPrimitive Interface="PLUG" Event="LD" Parameters="WO,WKPC" />
        <OutputPrimitive Interface="SOCKET" Event="LD" Parameters="WO,WKPC"
/>
      </ServiceTransaction>
      <ServiceTransaction >
        <InputPrimitive Interface="SOCKET" Event="UNLD" />
        <OutputPrimitive Interface="PLUG" Event="UNLD" />
      </ServiceTransaction>
      <ServiceTransaction >
        <InputPrimitive Interface="PLUG" Event="CNF" />
```

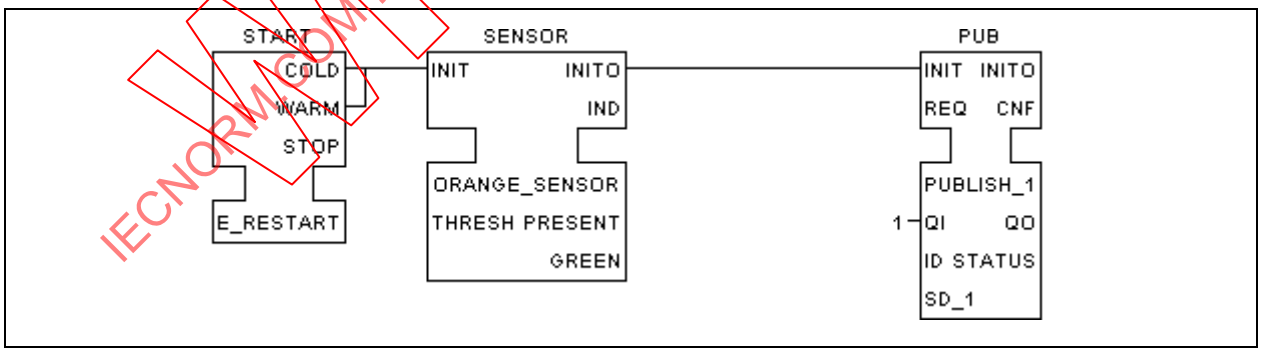
```
        <OutputPrimitive Interface="SOCKET" Event="CNF" />
      </ServiceTransaction>
    </ServiceSequence>
  </Service>
</AdapterType>
```

A.3.4 Resource types

EXAMPLE 1. A resource type containing an instance of the ORANGE_SENSOR function block type defined in A.3.2, plus an instance of the PUBLISH_1 type to transmit a change in condition, could be declared textually as shown below. This resource type also contains an instance of the E_RESTART type defined in Annex A of IEC 61499-1, interconnected to provide initialization of the other function block instances. The data outputs of the ORANGE_SENSOR block and the SD_1 input of the PUBLISH_1 block, and their corresponding event inputs and outputs, are left unconnected in order to allow application-specific logic to determine the value to be transmitted and the event to trigger the transmission.

```
RESOURCE_TYPE ORANGE_RES
     (* A Configurable Orange Presence&Quality Sensor *)
FB_TYPES
  E_RESTART ;
  ORANGE_SENSOR ;
  PUBLISH_1 ;
END_FB_TYPES
FBS
    START : E_RESTART;
    SENSOR : ORANGE_SENSOR;
    PUB : PUBLISH_1;
END_FBS
EVENT_CONNECTIONS
    START.COLD TO SENSOR.INIT ;
    START.WARM TO SENSOR.INIT ;
    SENSOR.INITO TO PUB.INIT ;
END_CONNECTIONS
DATA_CONNECTIONS
    1 TO PUB.QI ;
END_CONNECTIONS
END_RESOURCE_TYPE
```

A graphical representation of this resource's function block network is:



An equivalent XML document (with additional information for software tools) could be:

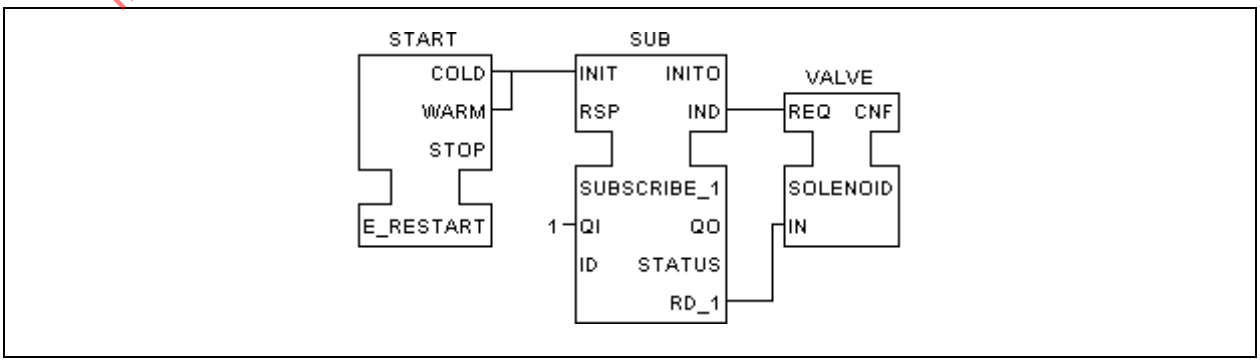
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ResourceType SYSTEM "../LibraryElement.dtd" >
<ResourceType Name="ORANGE_RES" Comment="A Configurable Orange
Presence&#38;Quality Sensor" >
  <Identification Standard="61915" Classification="C0202"
ApplicationDomain="Food Processing" Function="Detection" Type="Photoelectric
Sensors" Description="Orange Presence and Quality" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-06-20" Remarks="Corrected &#34;FBType&#34; to
```

```
&#34;FBTypeName&#34;." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-02-02" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <FBTypeName Name="E_RESTART" />
  <FBTypeName Name="ORANGE_SENSOR" />
  <FBTypeName Name="PUBLISH_1" />
  <FBNetwork >
    <FB Name="START" Type="E_RESTART" x="94.1176" y="11.7647" />
    <FB Name="SENSOR" Type="ORANGE_SENSOR" x="723.5294" y="11.7647" />
    <FB Name="PUB" Type="PUBLISH_1" x="2170.5881" y="11.7647" />
    <EventConnections>
      <Connection Source="START.COLD" Destination="SENSOR.INIT" dx1="35.2941"
dx2="47.0588" dy="-70.5882" />
      <Connection Source="START.WARM" Destination="SENSOR.INIT" dx1="58.8235"
dx2="76.4706" dy="-188.2353" />
      <Connection Source="SENSOR.INITO" Destination="PUB.INIT" dx1="170.5882"
/>
    </EventConnections>
    <DataConnections>
      <Connection Source="1" Destination="PUB.QI" dx1="2105.8823" />
    </DataConnections>
  </FBNetwork>
</ResourceType>
```

EXAMPLE 2. A resource type containing an instance of the SOLENOID function block type defined in A.3.2, plus an instance of the SUBSCRIBE_1 type to receive a command to change solenoid status, could be declared textually as shown below. This resource type also contains an instance of the E_RESTART type defined in Annex A of IEC 61499-1, interconnected to provide initialization of the other function block instances.

```
RESOURCE_TYPE SV_RESOURCE
    (* A Remotely Activated Solenoid Valve Resource *)
FBS
    START : E_RESTART;
    SUB : SUBSCRIBE_1;
    VALVE : SOLENOID;
END_FBS
EVENT_CONNECTIONS
    START.COLD TO SUB.INIT ;
    START.WARM TO SUB.INIT ;
    SUB.IND TO VALVE.REQ ;
END_CONNECTIONS
DATA_CONNECTIONS
    SUB.RD_1 TO VALVE.IN ;
    1 TO SUB.QI ;
END_CONNECTIONS
END_RESOURCE_TYPE
```

A graphical representation of this resource's function block network is:



An equivalent XML document (with additional information for software tools) could be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ResourceType SYSTEM "../LibraryElement.dtd" >
<ResourceType Name="SV_RESOURCE" Comment="A Remotely Activated Solenoid Valve
Resource" >
  <Identification Standard="61915" Classification="C0403"
ApplicationDomain="Any" Function="Logic / I/O Modules &#38; Controllers"
Type="Actuators" Description="Solenoid Valve" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-06-20" Remarks="Now uses LibraryElement.dtd" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-02-02" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <FBNetwork >
    <FB Name="START" Type="E_RESTART" x="70.5882" y="11.7647" />
    <FB Name="SUB" Type="SUBSCRIBE_1" x="705.8823" y="11.7647" />
    <FB Name="VALVE" Type="SOLENOID" x="1317.647" y="123.5294" />
    <EventConnections>
      <Connection Source="START.COLD" Destination="SUB.INIT" dx1="35.2941"
dx2="41.1765" dy="-64.7059" />
      <Connection Source="START.WARM" Destination="SUB.INIT" dx1="58.8235"
dx2="64.7059" dy="-182.3529" />
      <Connection Source="SUB.IND" Destination="VALVE.REQ" dx1="41.1765" />
    </EventConnections>
    <DataConnections>
      <Connection Source="SUB.RD_1" Destination="VALVE.IN" dx1="135.2941" />
      <Connection Source="1" Destination="SUB.QI" dx1="641.1765" />
    </DataConnections>
  </FBNetwork>
</ResourceType>
```

A.3.5 Device types

EXAMPLE 1. A device type containing an instance of the ORANGE_RES resource type defined in A.3.4 could be declared textually as shown below.

```
DEVICE_TYPE ORANGE_EYE
    (* Programmable Orange Presence+Quality Sensor *)
RESOURCE_TYPES
  ORANGE_RES ;
END_RESOURCE_TYPES
RESOURCE R1 : ORANGE_RES
END_RESOURCE
END_DEVICE_TYPE
```

An equivalent XML document (with additional information for software tools) could be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DeviceType SYSTEM "../LibraryElement.dtd" >
<DeviceType Name="ORANGE_EYE" Comment="Programmable Orange Presence+Quality
Sensor" >
  <Identification Standard="61915" Classification="C0202"
ApplicationDomain="Food Processing" Function="Detection"
Type="Photoelectric Sensors" Description="Orange Presence and Quality" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-06-20" Remarks="Corrected &#34;ResourceTypeName&#34;" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-02-02" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <ResourceTypeName Name="ORANGE_RES" />
  <Resource Name="R1" Type="ORANGE_RES" >
  </Resource>
</DeviceType>
```