



BSI Standards Publication

Programmable controllers

Part 10: PLC open XML exchange format (IEC 61131-10:2019)

National foreword

This British Standard is the UK implementation of EN IEC 61131-10:2019. It is identical to IEC 61131-10:2019.

The UK participation in its preparation was entrusted to Technical Committee GEL/65, Measurement and control.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2019
Published by BSI Standards Limited 2019

ISBN 978 0 580 97336 9

ICS 25.040.40; 35.240.30; 35.240.50

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 July 2019.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

English Version

Programmable controllers - Part 10: PLC open XML exchange
format
(IEC 61131-10:2019)

Automates programmables - Partie 10: Format d'échange
XML ouvert PLC
(IEC 61131-10:2019)

Speicherprogrammierbare Steuerungen - Teil 10: XML-
basiertes Austauschformat für Programme nach IEC 61131-
3
(IEC 61131-10:2019)

This European Standard was approved by CENELEC on 2019-05-29. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.



European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

European foreword

The text of document 65B/1147/FDIS, future edition 1 of IEC 61131-10, prepared by SC 65B "Measurement and control devices" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN IEC 61131-10:2019.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2020-02-29
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2022-05-29

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 61131-10:2019 was approved by CENELEC as a European Standard without any modification.

Annex ZA

(normative)

Normative references to international publications with their corresponding European publications

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE 1 Where an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 61131-1	-	Programmable controllers - Part 1: General information	EN 61131-1	-
IEC 61131-3	-	Programmable controllers - Part 3: Programming languages	EN 61131-3	-

This page deliberately left blank

CONTENTS

FOREWORD.....	8
INTRODUCTION.....	10
1 Scope.....	11
1.1 General.....	11
1.2 Implementation specific parameters	12
2 Normative references	13
3 Terms, definitions, abbreviated terms and acronyms	13
3.1 General terms and definitions	13
3.2 Abbreviated terms.....	13
4 Overview of schema concepts	14
4.1 Schema versioning	14
4.2 Naming conventions.....	14
4.3 Coordinate system of graphical languages.....	14
4.4 Schema extension concepts.....	17
5 Compliance	18
5.1 General.....	18
5.2 Feature tables.....	18
5.3 Vendor's compliance statement.....	18
6 Main schema element "Project"	19
6.1 General.....	19
6.2 "FileHeader"	19
6.3 "ContentHeader"	20
6.4 "Types"	21
6.5 "Instances"	21
6.5.1 General ("Configuration").....	21
6.5.2 "Resource"	22
6.5.3 "AccessVars"	25
6.5.4 "ConfigVars"	25
7 Abstract complex types.....	26
7.1 Purpose of abstract complex types	26
7.2 Abstract complex types for data type specifications	27
7.2.1 General	27
7.2.2 "TypeSpecBase".....	27
7.2.3 "InstantlyDefinableTypeSpecBase"	27
7.3 Abstract complex types for behaviour representations	27
7.3.1 General	27
7.3.2 "BehaviourRepresentationBase"	28
7.3.3 "ProgrammingLanguageBase"	28
7.4 Abstract complex types for graphical objects.....	28
7.4.1 General	28
7.4.2 "IdentifiedObjectBase".....	30
7.4.3 "GraphicalObjectBase"	30
7.4.4 "CommonObjectBase"	31
7.4.5 "FbdObjectBase"	31
7.4.6 "LdObjectBase"	31
7.4.7 "SfcObjectBase"	31

7.4.8	"NetworkBase"	32
7.5	Abstract complex types for textual constructs	33
7.5.1	General	33
7.5.2	"TextualObjectBase"	34
7.5.3	"NamespaceContentBase"	35
7.5.4	"TaskBase"	36
8	Namespace declaration	36
9	User-defined data type declaration	37
9.1	"UserDefinedTypeDecl"	37
9.2	"ArrayTypeSpec"	37
9.3	"DirectlyDerivedTypeSpec"	38
9.4	"EnumTypeSpec"	38
9.5	"EnumTypeWithNamedValueSpec"	39
9.6	"StructTypeSpec"	39
9.7	"SubrangeTypeSpec"	40
9.8	"ReferenceTypeSpec"	40
9.9	"ElementaryType"	41
10	POU declaration	41
10.1	"PouDecl"	41
10.2	"Program"	41
10.3	"FunctionBlock"	43
10.4	"Class"	44
10.5	"Function"	45
10.6	"Interface"	46
10.7	"Action"	46
10.8	"NamedTransition"	47
10.9	"MethodPrototype"	47
10.10	"Method"	48
10.11	"ParameterSet"	50
10.12	"VarListWithAccessSpec"	52
10.13	"AccessSpecifiers"	52
10.14	"Body"	52
10.15	"BodyWithoutSFC"	53
10.16	"Predicate"	53
11	Variable declaration	54
11.1	"VarList"	54
11.2	"ExternalVarList"	55
11.3	"VariableDecl"	55
11.4	"VariableDeclPlain"	56
11.5	"TypeRef"	56
11.6	"Value"	56
11.7	"AddressExpression"	57
11.8	"FixedAddressExpression"	58
12	Behaviour representation	58
12.1	"IL"	58
12.2	"ST"	58
12.3	"FBD"	59
12.4	"FbdNetwork"	59

12.5	"LD"	59
12.6	"LadderRung"	60
12.7	"SFC"	60
13	Graphical behaviour representation	60
13.1	General	60
13.2	Common elements	61
13.2.1	"Comment"	61
13.2.2	"Connector"	61
13.2.3	"Continuation"	62
13.2.4	"ActionBlocks"	62
13.3	FBD elements	64
13.3.1	"Block"	64
13.3.2	"graphicalFormalParameterCommon"	67
13.3.3	"DataSource"	67
13.3.4	"DataSink"	68
13.3.5	"Unconnected"	68
13.3.6	"Jump"	69
13.3.7	"Return"	70
13.4	LD elements	70
13.4.1	"LeftPowerRail"	70
13.4.2	"RightPowerRail"	71
13.4.3	"Coil"	71
13.4.4	"Contact"	72
13.4.5	"CompareContact"	73
13.5	SFC elements	74
13.5.1	"Step"	74
13.5.2	"Transition"	75
13.5.3	"SelectionDivergence"	76
13.5.4	"SelectionConvergence"	77
13.5.5	"SimultaneousDivergence"	78
13.5.6	"SimultaneousConvergence"	78
13.6	Connections	79
13.6.1	General	79
13.6.2	"ConnectionPointIn"	79
13.6.3	"Connection"	80
13.6.4	"FeedbackConnection"	81
13.6.5	"ConnectionPointOut"	81
14	Resource declaration	82
14.1	"StandardTask"	82
14.2	"ParameterAssignment"	82
15	Miscellaneous	82
15.1	"XyDecimalValue"	82
15.2	"AddData"	83
15.3	"TextBase"	83
15.4	"SimpleText"	83
15.5	"EdgeModifierType"	84
Annex A (normative)	Formal XML exchange format schema definition	85
Annex B (informative)	Recommended schemata	161

B.1	General.....	161
B.2	Recommended schemata to be used by "AddData"	164
B.3	Recommended schemata to be used by abstract complex type	172
Annex C (informative) Example XML document.....		190
Bibliography.....		276

Figure 1 – Main overview of XML exchange format usage (example)	11
Figure 2 – Mapping coordinate information to the coordinate system	15
Figure 3 – Transforming position using the scaling information	15
Figure 4 – Objects anchor points and object rectangles examples	17
Figure 5 – Main schema element "Project"	19
Figure 6 – Element "FileHeader"	20
Figure 7 – Element "ContentHeader"	20
Figure 8 – Element "Types"	21
Figure 9 – Element "Instances"	22
Figure 10 – Element "Resource"	23
Figure 11 – Element "ProgramInstance"	24
Figure 12 – Element "AccessVars"	25
Figure 13 – Element "ConfigVars"	26
Figure 14 – Extension relationship among complex types for data type specifications	27
Figure 15 – Extension relationship among complex types for behaviour representations	28
Figure 16 – Extension relationship among complex types for graphical objects	29
Figure 17 – Complex type "IdentifiedObjectBase"	30
Figure 18 – Complex type "GraphicalObjectBase"	30
Figure 19 – Complex type "CommonObjectBase"	31
Figure 20 – Complex type "FbdObjectBase"	31
Figure 21 – Complex type "LdObjectBase"	32
Figure 22 – Complex type "SfcObjectBase"	32
Figure 23 – Complex type "NetworkBase"	33
Figure 24 – Extension relationship among complex types for textual objects	34
Figure 25 – Complex type "TextualObjectBase"	35
Figure 26 – Complex type "NamespaceContentBase"	35
Figure 27 – Complex type "TaskBase"	36
Figure 28 – Complex type "NamespaceDecl"	36
Figure 29 – Complex type "UserDefinedTypeDecl"	37
Figure 30 – Complex type "ArrayTypeSpec"	38
Figure 31 – Complex type "DirectlyDerivedTypeSpec"	38
Figure 32 – Complex type "EnumTypeSpec"	39
Figure 33 – Complex type "EnumTypeWithNamedValueSpec"	39
Figure 34 – Complex type "StructTypeSpec"	40
Figure 35 – Complex type "SubrangeTypeSpec"	40
Figure 36 – Complex type "ReferenceTypeSpec"	40
Figure 37 – Complex type "PouDecl"	41

Figure 38 – Complex type "Program"	42
Figure 39 – Complex type "FunctionBlock"	43
Figure 40 – Complex type "Class"	44
Figure 41 – Complex type "Function"	45
Figure 42 – Complex type "Interface"	46
Figure 43 – Complex type "Action"	46
Figure 44 – Complex type "NamedTransition"	47
Figure 45 – Complex type "MethodPrototype"	48
Figure 46 – Complex type "Method"	49
Figure 47 – Complex type "ParameterSet"	51
Figure 48 – Complex type "VarListWithAccessSpec"	52
Figure 49 – Complex type "Body"	53
Figure 50 – Complex type "BodyWithoutSFC"	53
Figure 51 – Complex type "Predicate"	54
Figure 52 – Complex type "VarList"	54
Figure 53 – Complex type "ExternalVarList"	55
Figure 54 – Complex type "VariableDecl"	55
Figure 55 – Complex type "VariableDeclPlain"	56
Figure 56 – Complex type "TypeRef"	56
Figure 57 – Complex type "Value"	57
Figure 58 – Complex type "AddressExpression"	57
Figure 59 – Complex type "FixedAddressExpression"	58
Figure 60 – Complex type "IL"	58
Figure 61 – Complex type "ST"	58
Figure 62 – Complex type "FBD"	59
Figure 63 – Complex type "FbdNetwork"	59
Figure 64 – Complex type "LD"	59
Figure 65 – Complex type "LadderRung"	60
Figure 66 – Complex type "SFC"	60
Figure 67 – Complex type "Comment"	61
Figure 68 – Complex type "Connector"	61
Figure 69 – Complex type "Continuation"	62
Figure 70 – Complex type "ActionBlocks"	63
Figure 71 – Complex type "Block"	66
Figure 72 – Complex type "DataSource"	67
Figure 73 – Complex type "DataSink"	68
Figure 74 – Complex type "Unconnected"	69
Figure 75 – Complex type "Jump"	69
Figure 76 – Complex type "Return"	70
Figure 77 – Complex type "LeftPowerRail"	70
Figure 78 – Complex type "RightPowerRail"	71
Figure 79 – Complex type "Coil"	72
Figure 80 – Complex type "Contact"	73

Figure 81 – Complex type "CompareContact"	74
Figure 82 – Complex type "Step"	75
Figure 83 – Complex type "Transition"	76
Figure 84 – Complex type "SelectionDivergence"	77
Figure 85 – Complex type "SelectionConvergence"	78
Figure 86 – Complex type "SimultaneousDivergence"	78
Figure 87 – Complex type "SimultaneousConvergence"	79
Figure 88 – Complex type "ConnectionPointIn"	80
Figure 89 – Complex type "Connection"	80
Figure 90 – Complex type "FeedbackConnection"	81
Figure 91 – Complex type "ConnectionPointOut"	81
Figure 92 – Complex type "StandardTask"	82
Figure 93 – Complex type "ParameterAssignment"	82
Figure 94 – Complex type "XyDecimalValue"	83
Figure 95 – Complex type "AddData"	83
Figure 96 – Complex type "TextBase"	83
Figure 97 – Complex type "SimpleText"	84
Figure B.1 – Only IEC 61131-3 features	161
Figure B.2 – Vendor specific extensions "AddData"	162
Figure B.3 – Vendor specific extensions (abstract complex type)	163

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PROGRAMMABLE CONTROLLERS –**Part 10: PLC open XML exchange format****FOREWORD**

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of 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, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). 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. 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 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 IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61131-10 has been prepared by subcommittee 65B: Measurement and control devices, of IEC technical committee 65: Industrial-process measurement, control and automation.

The text of this International Standard is based on the following documents:

FDIS	Result on voting
65B/1147/FDIS	65B/1153/RVD

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 61131 series, published under the general title *Programmable controllers*, can be found on the IEC website.

This IEC standard includes Code Components i.e. components that are intended to be directly processed by a computer. Such content is any text found between the markers <CODE BEGINS> and <CODE ENDS>, or otherwise is clearly labelled in this standard as a Code Component.

The purchase of this IEC standard carries a copyright license for the purchaser to sell software containing Code Components from this standard to end users either directly or via distributors, subject to IEC software licensing conditions, which can be found at: www.iec.ch/CCv1.

The Code Components included in this IEC standard are also available as an electronic machine-readable file at <http://www.plcopen.org/technical-activities/IEC61131-10/CodeComponents/PLCopenXML.htm>.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The International Standard IEC 61131 describes programmable logic controllers (PLCs).

IEC 61131-3 defines programming languages. Users want standardized programming languages and the ability to exchange a complete program or parts of that program between different development environments, i.e. from an exporting environment to an importing environment.

IEC 61131-3 defines program organization units (POUs). But an entire program also consists of user-defined data types, global and external declarations and other elements besides the POUs. In this document, the term "IEC 61131-3 project" is used. It contains all above-mentioned language elements, required for an exchange, in order to get a consistent program in the importing environment.

The exchange of POUs developed in one of the textual languages, i.e. instruction list (IL) and structured text (ST) or the textual representation of sequential function charts (SFC) is possible, because a syntax description of these languages is part of the IEC 61131-3 standard. The objective of this document is to extend the reuse of programmed solutions both for textual languages and graphical languages, i.e. function block diagram (FBD) and ladder diagram (LD) or the graphical representation of SFCs. Furthermore, the completeness of exchange between the different environments depends on the supported features that are listed in the compliance list defined in IEC 61131-3.

This document defines a solution independent eXtensible Markup Language (XML) based exchange format, to be supported by interfaces of different kinds of software tools. Beside textual and program logic information, it also provides the ability to transfer graphical representation information, e.g. the position and size of function blocks and how they are connected. The design of the 'transferred' parts shall represent the same program logic, however it may be altered in look and feel.

This document's XML exchange format enables a transfer of IEC 61131-3 projects, from an exporting environment to an importing environment, including extensions for layout and formatting.

This document's XML exchange format can not only describe correct IEC 61131-3 POUs, but it can represent a working state of the IEC 61131-3 project. For example, even if the IEC 61131-3 source project is incomplete, for example if it contains compile errors, it can be represented.

Syntactically incorrect IEC 61131-3 projects can be represented. For example, such a project could be an in-between version or a project containing several unconnected FBD blocks.

This document's XML exchange format provides for life cycle management of automation systems, e.g. in case of redesign, maintenance or device replacement. If an IEC 61131-3 project is stored in this standard's XML exchange format, it could be reused independent of a special development environment. And thus, it could be modified and maintained by any other development environment supporting this standard's XML exchange format.

This International Standard was developed using material from PLCopen^{®1}. This document extends PLCopen[®] XML, adopts it to the features of IEC 61131-3:2013 and is therefore not compatible with previous versions of PLCopen[®] XML.

¹ PLCopen[®] is the registered trademark of PLCopen. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

PROGRAMMABLE CONTROLLERS –

Part 10: PLC open XML exchange format

1 Scope

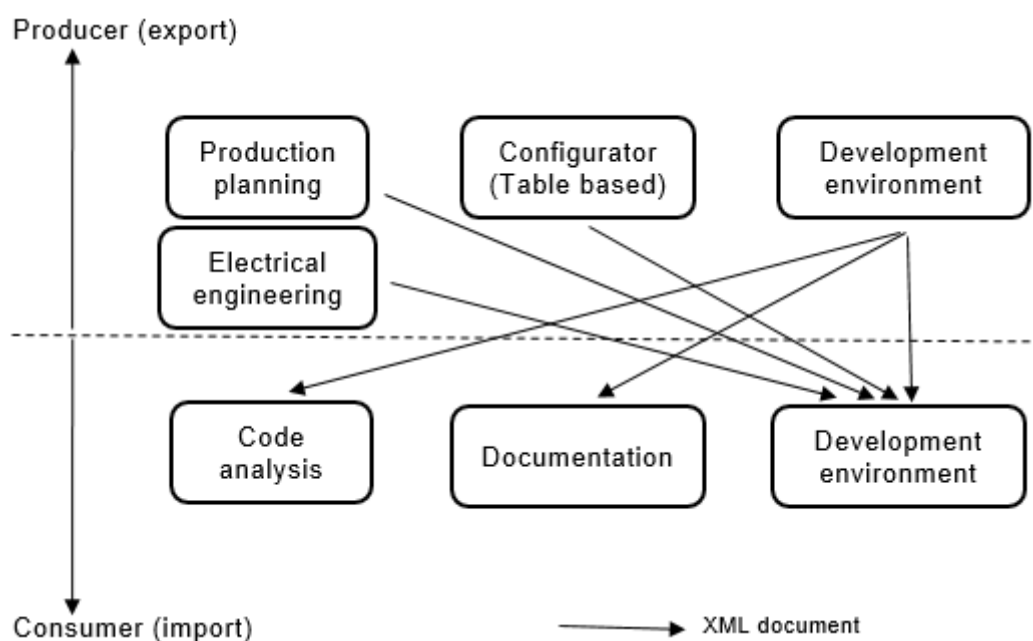
1.1 General

This part of IEC 61131 specifies an XML-based exchange format for the export and import of IEC 61131-3 projects. A complete IEC 61131-3 project implemented in an IEC 61131-3 environment can be transferred between different programming environments. It allows for the exchange of configuration elements, data types, and POU's written in:

- the textual language, instruction list (IL),
- the textual language, structured text (ST),
- the graphical language, ladder diagram (LD),
- the graphical language, function block diagram (FBD), and
- sequential function chart (SFC).

The exchange format is specified as a corresponding XML schema. The XML schema is an independent file with the .xsd extension and as such part of this specification. The specification of this schema is contained in Annex A. Annex B provides recommended schemata for extensions. An example XML document is given in Annex C. It is assumed that the reader of this document is familiar with XML technology.

Figure 1 provides an example overview of the usage of the XML exchange format. Different tools may produce and consume XML based IEC 61131-3 information.



IEC

Figure 1 – Main overview of XML exchange format usage (example)

The usage of the XML exchange format should provide more than a simple export/import from one development environment to another. All relevant information should be exported. This may include coordinate information for graphical tools. The importing tool should be able to filter which parts of this information need to be imported into its destination environment. Vendor-specific information and attributes may be included in the export file and selectively imported, if applicable. The vendor-specific information shall not influence the logic part of the program. Filtering should be done on the import – thus vendors shall ensure that their extensions of the XML schema are done in such a way that neglecting the information during import does not affect the functionality of the IEC 61131-3 project. Vendor specific attributes and information may be added by vendor specific XML schema – besides the XML exchange format defined in this document.

The described formats are designed for the import and export of IEC 61131-3 projects. Such an IEC 61131-3 project can be under development and as a consequence be incomplete.

Concerning the exchange of graphical language constructs between different programming systems, the focus is on logical information with optional explicit graphics.

1.2 Implementation specific parameters

This document does not provide means or requirements for compliant functionality (e.g. functional subset which has to be supported by all Programming and Debugging Tools (PADTs)). This document enables the exchange of all possible features defined in IEC 61131-3. Moreover, many implementation-specific features can be expressed using the AddData mechanism.

In some use cases, programs are either transferred from one PADT to another or generated for the use in a different PADT. In both cases, the function set of these PADTs may be different as well as their settings of implementation-dependent parameters. If several PADTs have to be supported/considered, the functionality of the program has to be restricted to the subset supported by all PADTs in question. Some of these functions can be determined from the IEC 61131-3 feature tables of the concerned PADT, for example:

- supported data types and standard functions,
- pre-emptive or non-pre-emptive scheduling,
- SFC with or without a final scan, etc.

Other functions and settings of implementation dependent parameters may require more effort to determine, for example:

- maximum amounts of code or variables per POU,
- maximum length of identifiers (variable name length),
- size of STRING and WSTRING variables with default length or maximum length,
- SFC to evaluate all transition conditions or only those with active steps as predecessors,
- range and precision of data types TIME, DATE, TOD, DT,
- runtime performance of (the POU in) the PLC,
- execution order within a graphical network, etc.

These differences have to be considered for use cases with more than one PADT. In some cases it may be appropriate to use only functionality supported by all concerned PADTs; in other cases, it may be necessary to manually change and test the program after importing into the PADT.

This document does not state requirements regarding compliant functions of the PADT. It defines an exchange format to exchange programs that are compliant with IEC 61131-3.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61131-1, *Programmable controllers – Part 1: General information*

IEC 61131-3, *Programmable controllers – Part 3: Programming languages*

3 Terms, definitions, abbreviated terms and acronyms

3.1 General terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61131-3 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1.1

IEC 61131-3 project

set of information in accordance with IEC 61131-3 comprising POU types and instances, data types, global variables instances, configurations and resources

3.1.2

abstract complex type

refers to a "complexType" element from the XML schema definition, which has additionally the "abstract" attribute set to true

3.1.3

concrete complex type

refers to a "complexType" element from the XML Schema Definition which usually extends an abstract complex type without being abstract itself

3.1.4

simple type

refers to a "simpleType" element from the XML Schema Definition

3.2 Abbreviated terms

FBD	function block diagram
IL	instruction list
LD	ladder diagram
PLC	programmable logic controller
POU	program organization unit
PADT	programming and debugging tool
SFC	sequential function chart
ST	structured text
URI	uniform resource identifier
URL	uniform resource locator
XML	eXtensible Markup Language

4 Overview of schema concepts

4.1 Schema versioning

The schema contains a version attribute. Its first number equals to the IEC 61131-10 edition number. The second number is reserved for minor changes within one edition. The schema file name reflects this version too.

Additionally, there is a "schemaVersion" attribute indicating which schema version an XML document was created with.

4.2 Naming conventions

Within all defined schemata, including the recommendations, the following naming conventions are used:

- attribute names shall start with lower case letter;
- type and element names shall start with upper case letter;
- names consisting of multiple words shall have the first character of each word, except the first, starting with a capital letter. No underscores shall be used.

4.3 Coordinate system of graphical languages

This document supports the exchange of IEC 61131-3 projects including POU's programmed in one of the graphical languages, i.e. FBD, LD and the graphical representation of SFC elements. This document defines a virtual coordinate system in order to optionally express the arrangement of graphical elements (e.g. function block, contact, coil, and step). This enables (but does not require) the importing system to arrange the graphical representation of the imported code as similarly as possible to its representation in the exporting system. Therefore, each element in a graphical language can store its relative position and size as an XML attribute. Additionally, a sub-element (e.g. formal parameters of a function block invocation) can store a relative position relative to its parent element. Furthermore, a connection between two elements can store a set of relative positions to express the routing layout. Connections are described in 13.6.

The relative position and size are optional attributes and shall be used for graphical representation only. The relative position is not an indication of execution order.

The virtual coordinate system has two axes X and Y. The origin (0,0) of the coordinate system shall be the top-left corner of the graphical body. Positive X is to the right, and positive Y is down. A relative position of a graphical element identifies the upper left corner of this element. Consequently, a relative position between a sub-element and its parent may have positive or negative values.

In order to help the importing system to scale the graphics, the exporting system can add information about its usage of the virtual coordinate system in the "CoordinateInfo" of the "ContentHeader" as shown in Figure 2.

- For FBD – the minimum distance between two pins, both for X and Y coordinates
- For LD – the size of a coil (width for X and height for Y)
- For SFC – the size of a transition (width for X and height for Y)

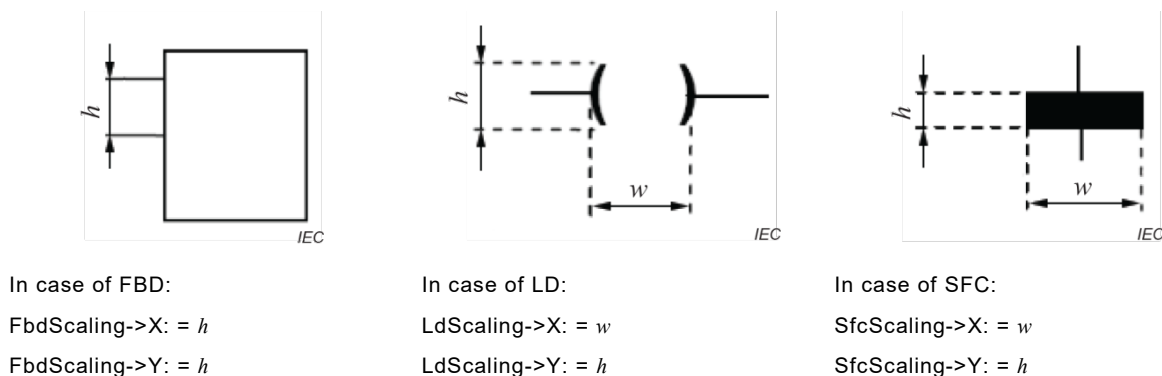


Figure 2 – Mapping coordinate information to the coordinate system

With this given information, a coordinate transformation can be performed. For example, if a system A with FbdScaling = 10 exports a function block invocation at the relative position P(30,20), then, during import, a system B with FbdScaling = 5 may transform the relative position P to (15,10) as shown in Figure 3.

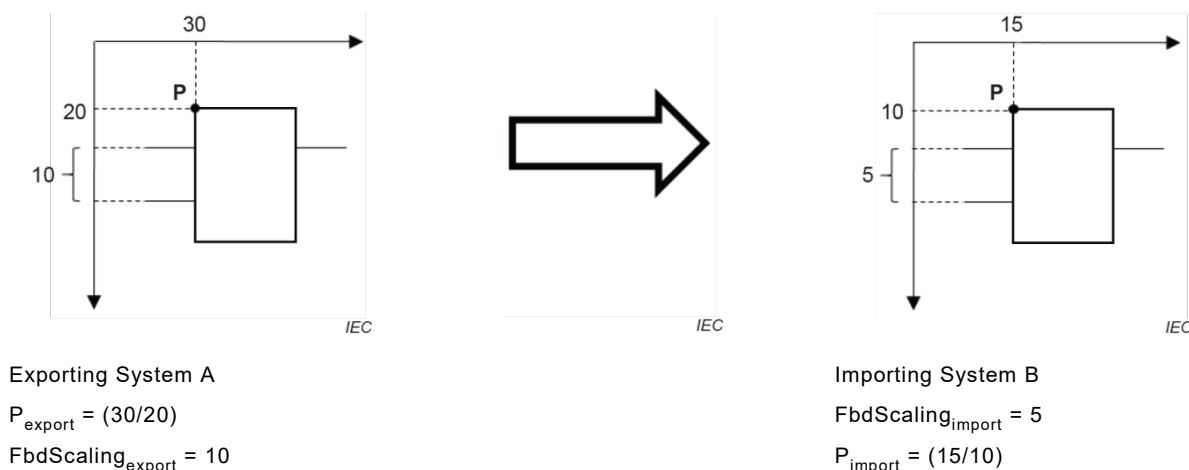
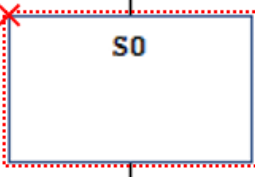
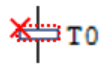
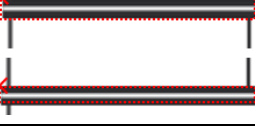
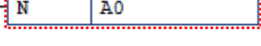
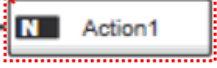
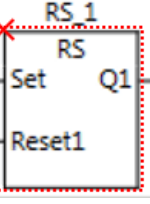
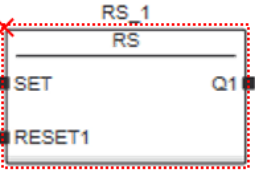
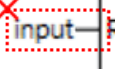
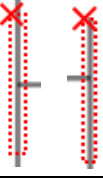


Figure 3 – Transforming position using the scaling information

The "RelPosition" element of a graphical object specifies the position of the object's anchor point. "RelPosition" is of type "XyDecimalValue" and has the attributes "x" and "y". The anchor point of an object shall be the upper left corner of the object rectangle. The object rectangle contains only the main body of the object. Attached elements like labels (instance name, coil name) or negations shall not be considered for the size of this rectangle. The size of the object rectangle shall be specified by the "Size" element (also of type "XyDecimalValue") of the graphical object, where the width corresponds to "x" and the height corresponds to "y". Examples of graphical elements are given in Figure 4.

"RelPosition" always specifies relative position to the superior graphical object, whose type is derived from complexType "GraphicalObjectBase" or "BehaviourRepresentationBase":

- In the case of FBD, the "RelPosition" of a graphical object is related to its superior "FbdNetwork".
- In the case of LD, the "RelPosition" of a graphical object is related to its superior "Rung".
- The "RelPosition" of "FbdNetwork" and "Rung" is related to their superior "FBD" and "LD" element, respectively. In other words, this is the absolute position within the body.
- In the case of SFC, the "RelPosition" of graphical objects is related to its superior "SFC" element. In other words, this is the absolute position within the body.
- The "RelPosition" of "ConnectionPointIn" and "ConnectionPointOut" is related to its superior graphical element.
- "RelPosition" elements of "Connection" are related to its superior "ConnectionPointIn" element.

Object	Example 1	Example 2
step		
transition		
selection divergence / convergence		
simultaneous divergence / convergence		
action block		
block		
in/inout/out variable		
connector, continuation		
jump, label, return		
left/right power rail		

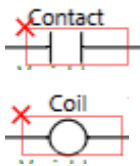
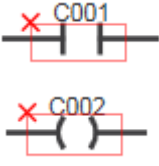
Object	Example 1	Example 2
contact, coil		

Figure 4 – Objects anchor points and object rectangles examples

4.4 Schema extension concepts

The schema is created in accordance with the language elements and features defined in IEC 61131-3. Nevertheless, a PADT using the schema might support additional features or might need to store additional information in the exported file. The schema supports two extension mechanisms allowing incorporation of information in the exchange format, which is outside the scope of IEC 61131-3.

The first mechanism is the "AddData" element, which can be nested below various other elements. Its usage is intended for the extension of other elements, by which they are enriched with information, but which is not crucial for the element's exchange between tools. Clause B.1 contains recommendations for common extensions, which can be done by using "AddData". They should be seen as a guideline for designing new extensions.

A PADT making use of the "AddData" mechanism should also provide the schemas defining those extensions. All additional schemas used shall be listed in the "AddDataInfo" element described in 6.3. Details about the contents of "AddData" are contained in 15.2.

The second mechanism usable for extensions is abstract complex types. All abstract complex types used in the schema are listed in Clause 7. Abstract complex types allow complex extensions of the schema which can be standalone and do not rely on extending already existing elements. Furthermore, IEC 61131-3 does not assertively limit available programming languages for PLCs only to IL, ST, FBD, LD or SFC. In this schema, the following elements are extensible by specifying abstract complex types for the elements:

- user-defined data types (by extending "TypeSpecBase");
- behaviour representations for body, activity, etc. (by extending "BehaviourRepresentationBase");
- graphical object types (by extending "CommonObjectBase", "FbdObjectBase", "LdObjectBase", or "SfcObjectBase");
- graphical expression for the condition of a transition in SFC (by extending "NetworkBase")
- task types (by extending "TaskBase");
- text for documentation (by extending "TextBase").

A new element type can be derived from the abstract complex type and used as a substitution for the element type where the abstract complex type is specified. Examples for such extensions are vendor-specific programming languages or task types. As with "AddData", the schemas for this extensions should also be made available for other tools required to understand the extensions. Clause B.2 contains a recommendation for common extensions, which can be done by using abstract complex types. It should be seen as a guideline for designing new extensions.

5 Compliance

5.1 General

A PLC programming and debugging tool (PADT), as defined in IEC 61131-1, which claims to comply, wholly or partially, with the requirements of this document shall:

- a) provide a subset of the features and provide the corresponding vendor's compliance statement as defined below;
- b) not require the inclusion of substitute or additional schema elements in order to accomplish any of the features;
- c) provide a document that specifies all vendor specific dependencies. This includes the implementation dependencies explicitly designated in this document and the limiting parameters such as maximum length, number, size and range of value, which are not explicitly defined in this document;
- d) provide a document that specifies all errors that are detectable and reported by the implementation;
- e) not use any of the schema element names defined in this document for implementation-defined features whose functionality differs from that described in this document.

5.2 Feature tables

This part re-uses the feature tables defined in IEC 61131-3. All tables in IEC 61131-3 are used for a special purpose in a common way. The first column contains the "feature number", the second column gives the "feature description", and the following columns may contain examples or further information. This table structure is used in the vendor's compliance statement.

There are 2 possibilities for a vendor in order to re-use the IEC 61131-3 feature tables for the purpose of this document:

- 1) adding 2 columns (XML export, XML import) to the feature tables of the PADT, or
- 2) creating a separate list of feature tables for the export/import functionality.

5.3 Vendor's compliance statement

The vendor may define any consistent subset of the features listed in the feature tables of IEC 61131-3 and shall declare the provided subset in the "Vendor's compliant statement".

The vendor's compliance statement shall be included in the documentation accompanying the system, or shall be produced by the system itself.

The format of the vendor's compliance statement shall provide the following information:

- The general information including the vendor name and address, the product name and version, and the date of issue.
- For each implemented feature, the number of the corresponding feature table, the feature number and the applicable programming language.

Optional is the title and subtitle of the feature table, the feature description, examples, vendor's note etc.

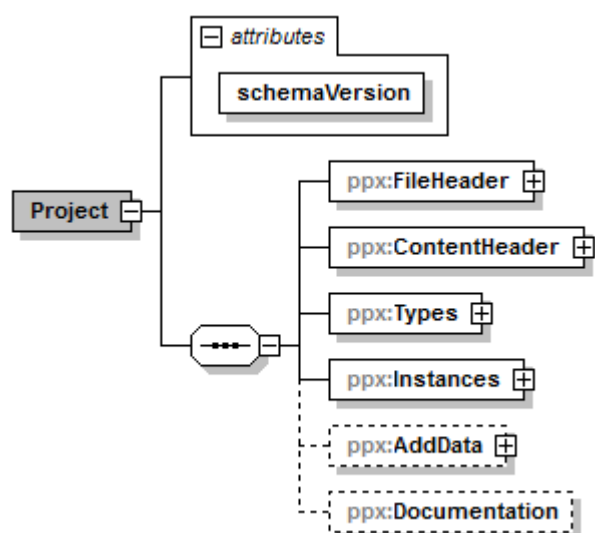
Tables and features not implemented may be omitted.

IEC 61131-3 provides an example of a compliance statement, which could be modified for use with this document.

6 Main schema element "Project"

6.1 General

The main schema element "Project" represents an IEC 61131-3 project. Its content is shown in Figure 5.



IEC

Figure 5 – Main schema element "Project"

The "schemaVersion" attribute indicates which schema version an XML document was created with.

"FileHeader", "ContentHeader", "Types", and "Instances" are explained in 6.2, 6.3, 6.4, and 6.5.

"Documentation" is of type "TextBase" which is explained in 15.3.

NOTE Figures in this document can contain rectangles constructed with solid or dashed lines. Solid lines indicate required information. Dashed lines indicate optional information, e.g. in Figure 5: ppx:FileHeader (solid lined rectangle) is required information and ppx:AddData (dashed rectangle) is optional information.

The figures are constructed using a XML engineering development environment, such as those listed in the Bibliography.

6.2 "FileHeader"

"FileHeader" represents information concerning the exporting tool of the XML file. Its content is shown in Figure 6.

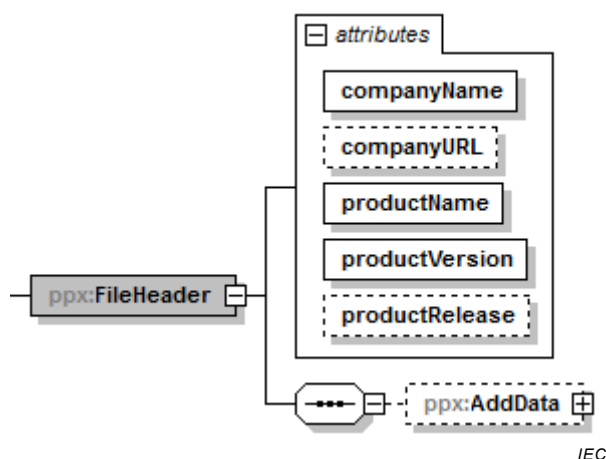


Figure 6 – Element "FileHeader"

The attributes "companyName" and "companyURL" are describing the vendor of the exporting tool, "productName", "productVersion", and "productRelease" the tool itself.

6.3 "ContentHeader"

"ContentHeader" represents overview information concerning the content of the XML file. Its content is shown in Figure 7.

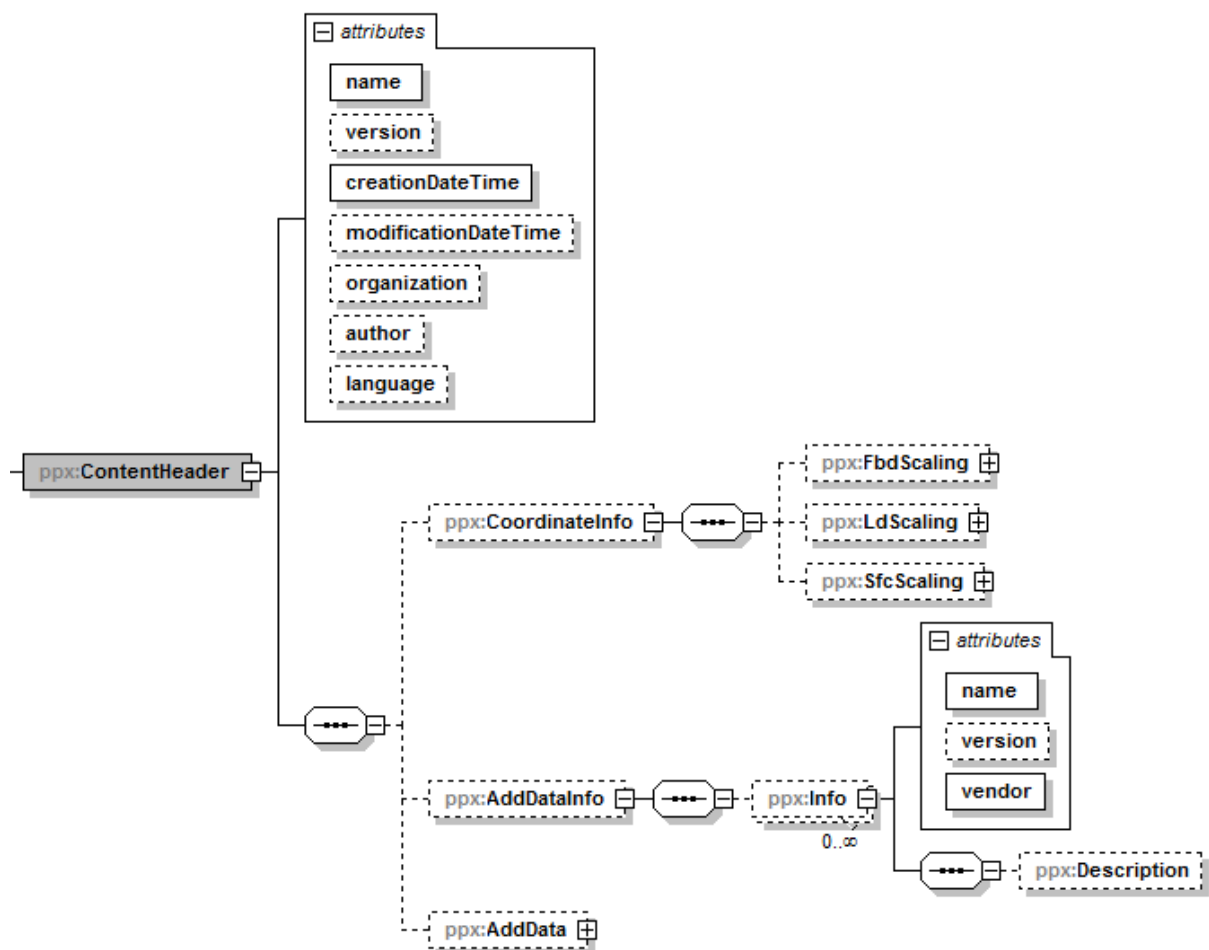


Figure 7 – Element "ContentHeader"

The attributes "name" (at export this is set to the project name), "version", "creationDateTime", "modificationDateTime", "organization", "author", and "language" (language used in the definition of the project) are available.

"FbdScaling", "LdScaling", and "SfcScaling" are of type "XyDecimalValue" which is explained in 15.1. The scaling concept is explained in 4.3.

The "AddDataInfo" represents a unique identification of the corresponding "AddData" content. The "name" attribute shall include the vendor domain (URI) to ensure unique names. The attribute "name" shall be used further in the produced XML document to refer to the corresponding AddData schema. Therefore "name" shall be unique in the produced XML document. The attribute "vendor" shall include the domain URL. Using this attribute "name" the importing tool may process the "AddData". The vendor shall specify the behaviour of the importing tool in case the name is not known by the importing tool especially regarding a later export from this tool. The attribute "version" indicates a schema version. "Description" is a formatted text according to <http://www.w3.org/1999/xhtml> and contains a brief description of each schema used in the file.

6.4 "Types"

"Types" represents all type related elements, as defined in IEC 61131-3. Its content is shown in Figure 8.

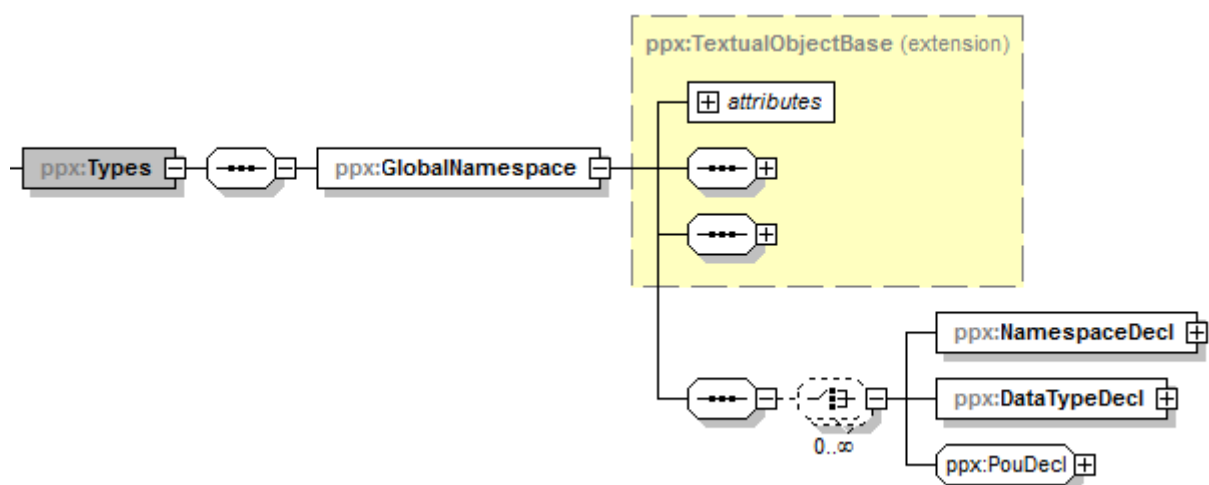


Figure 8 – Element "Types"

A "GlobalNamespace" is based on the abstract complex type "TextualObjectBase" which is explained in 7.5.2.

"NamespaceDecl" of type "NamespaceDecl" is explained in Clause 8. "DataTypeDecl" of type "UserDefinedTypeDecl" is explained in 9.1. The contents of the group "PouDecl" is explained in 10.1.

6.5 "Instances"

6.5.1 General ("Configuration")

"Instances" represents all instance related elements, as defined in IEC 61131-3. Its content is shown in Figure 9.

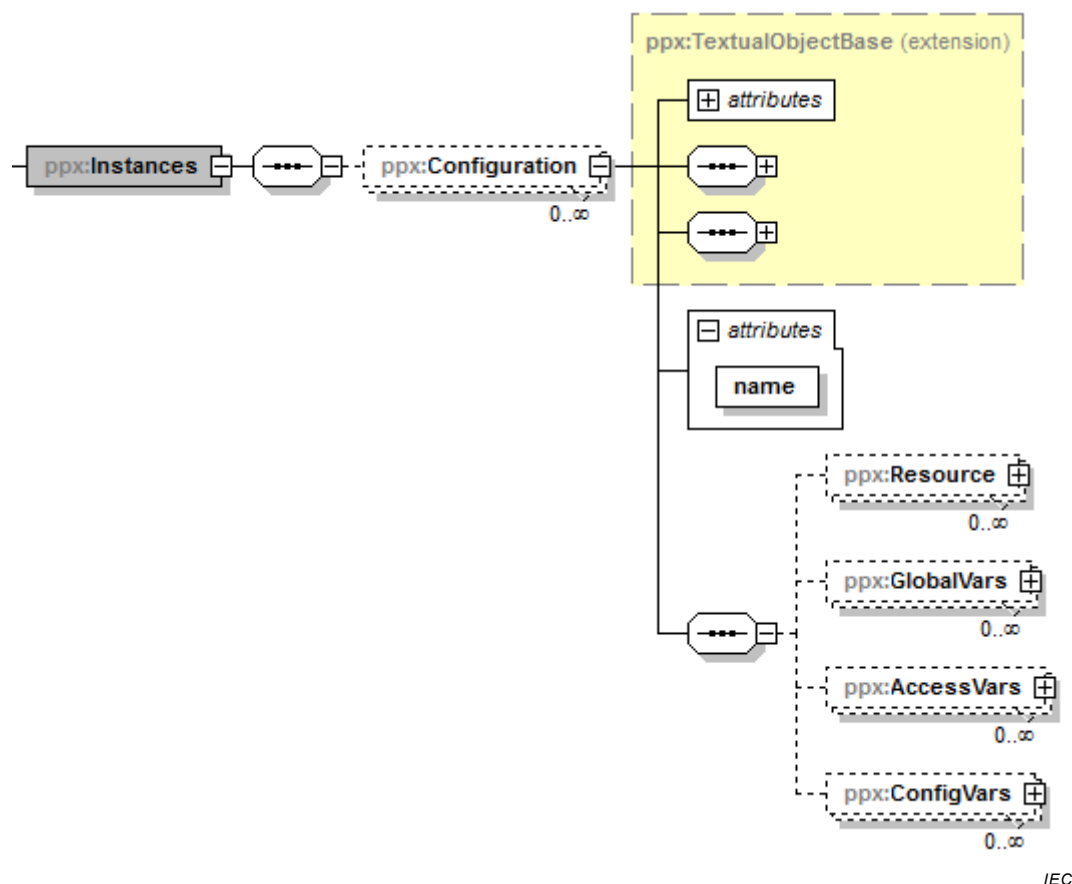


Figure 9 – Element "Instances"

A "Configuration" is based on the abstract complex type "TextualObjectBase" which is explained in 7.5.2. A "Configuration" is identified by the attribute "name".

"Resource", "GlobalVars" (IEC 61131-3: global variables) of type "VarList", "AccessVars", and "ConfigVars" are explained in 6.5.2, 11.1, 6.5.3, and 6.5.4.

6.5.2 "Resource"

"Resource" represents resource, as defined in IEC 61131-3. Its content is shown in Figure 10.

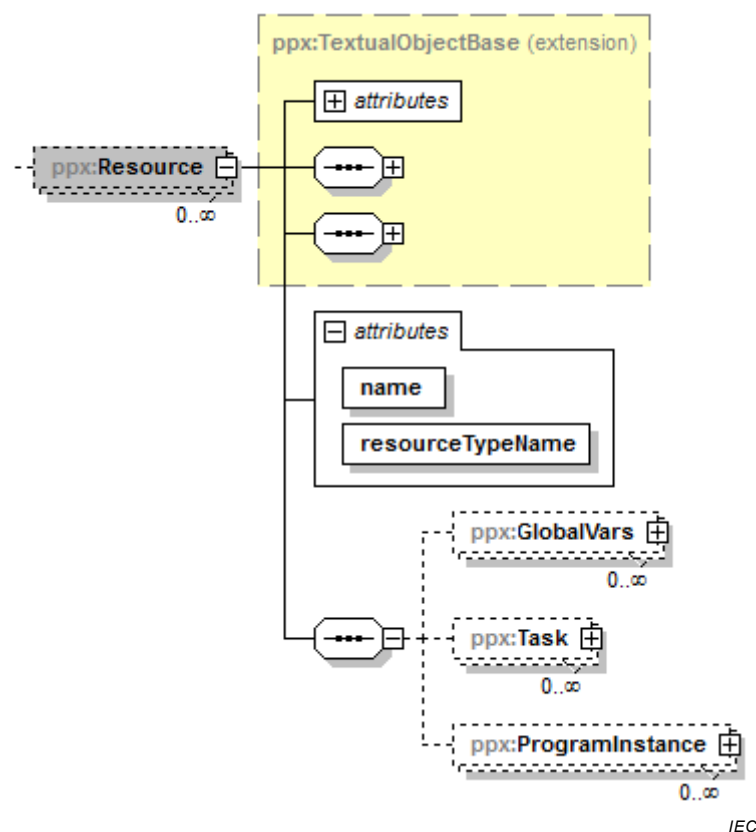


Figure 10 – Element "Resource"

A "Resource" is based on the abstract complex type "TextualObjectBase" which is explained in 7.5.2. A "Resource" is identified by the attribute "name". Additionally, the "resourceTypeName" is available.

A "Resource" contain "Task" of type "TaskBase" which is explained in 7.5.4, "GlobalVars" (IEC 61131-3: global variables) of type "VarList" which is explained in 11.1, and "ProgramInstance".

"ProgramInstance" represents program instance, as defined in IEC 61131-3. Its content is shown in Figure 11.

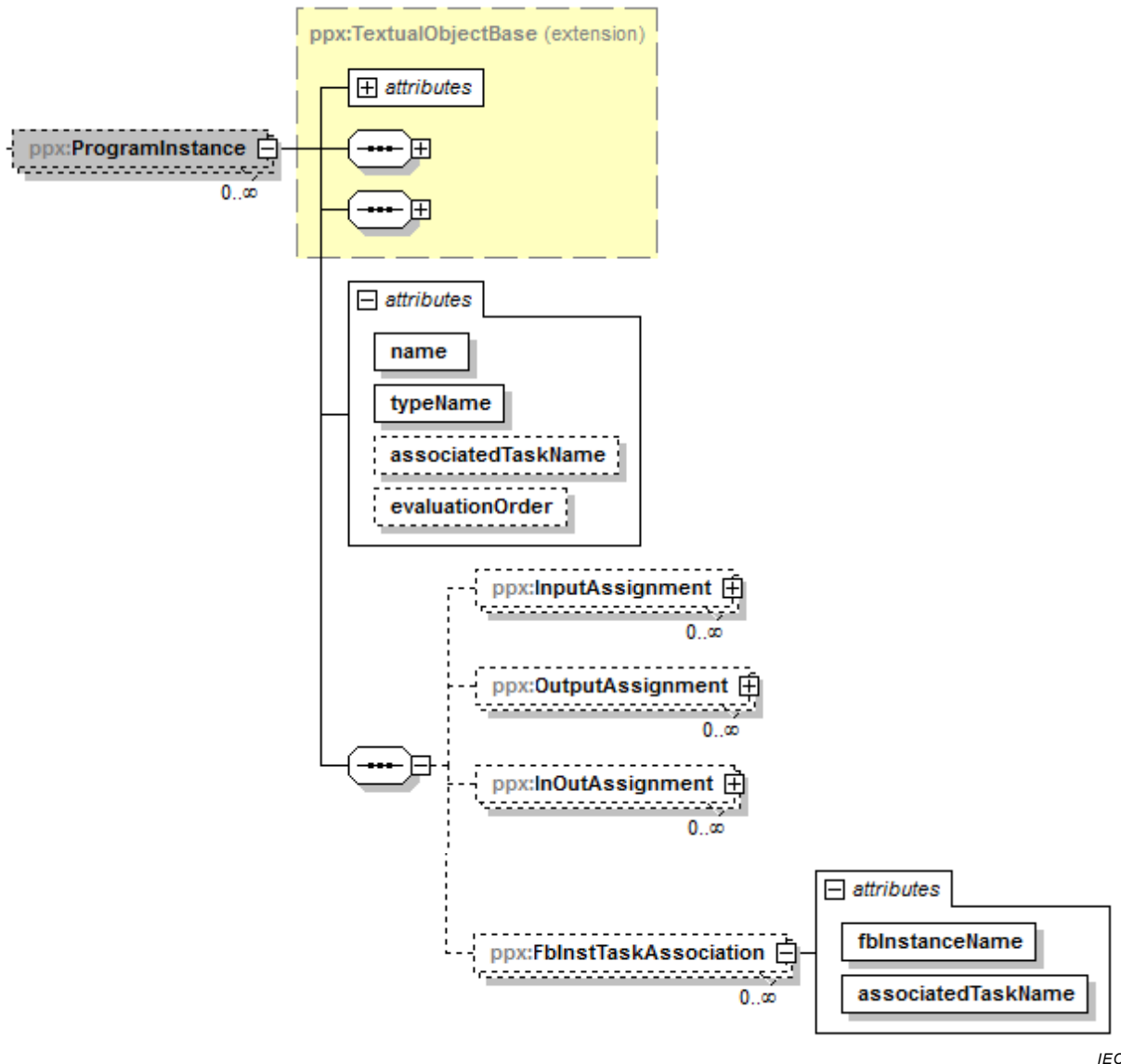


Figure 11 – Element "ProgramInstance"

A "ProgramInstance" is based on the abstract complex type "TextualObjectBase", which is explained in 7.5.2. A "ProgramInstance" is identified by the attribute "name". The name of the program instance shall be unique per resource.

The attribute "typeName" represents the program type name of this instance.

The attribute "associatedTaskName" represents the task name which this program instance is associated with. In the case this attribute is omitted, system-defined default task (cyclic task with lowest priority) shall be associated, as defined in IEC 61131-3. Within this priority it is evaluated according to its "evaluationOrder".

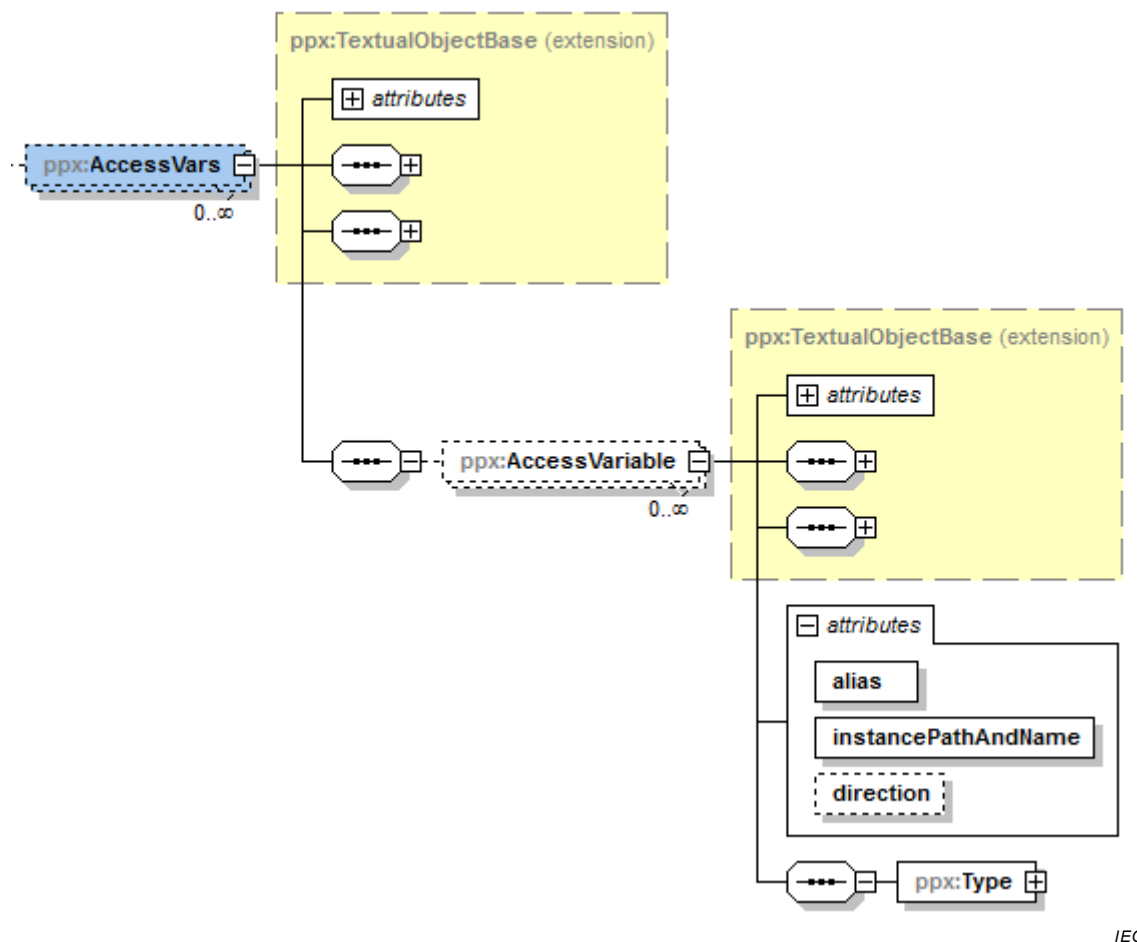
The attribute "evaluationOrder" represents the evaluation order within a task. The program instance with the smaller number of "evaluationOrder" shall be evaluated earlier. It shall not have negative value. This attribute is provided because XML 1.0 specification of W3C does not ensure that appearance order of sibling elements is significant. It is strongly recommended that the exporter should give ascending incremental number to the attribute of each "ProgramInstance" element per each task.

"InputAssignment", "OutputAssignment", and "InOutputAssignment" are of type "ParameterAssignment" which is explained in 14.2.

"FblInstanceTaskAssociation" represents the explicit association of a function block instance to a task. The attributes "fbInstanceName" and "associatedTaskName" are of the type string.

6.5.3 "AccessVars"

"AccessVars" represents access variables, as defined in IEC 61131-3. Its content is shown in Figure 12.



IEC

Figure 12 – Element "AccessVars"

"AccessVars" is based on the abstract complex type "TextualObjectBase" which is explained in 7.5.2. It consists of "AccessVariable", again based on "TextualObjectBase".

The attributes "alias", "instancePathAndName", and "direction" are available.

"Type" of type "TypeRef" is explained in 11.5.

6.5.4 "ConfigVars"

"ConfigVars" represents config variables, as defined in IEC 61131-3. Its content is shown in Figure 13.

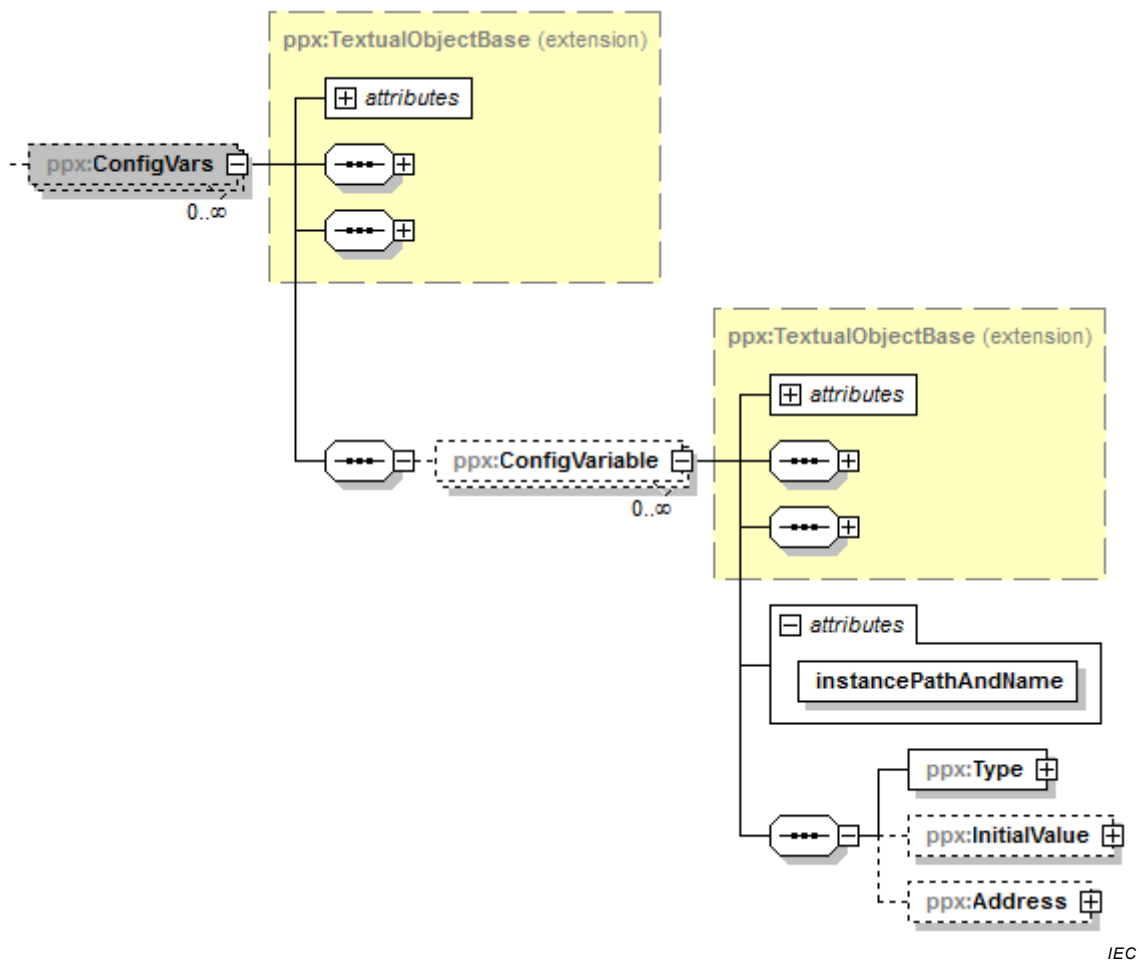


Figure 13 – Element "ConfigVars"

"ConfigVars" is based on the abstract complex type "TextualObjectBase" which is explained in 7.5.2. It consists of "ConfigVariable", again based on "TextualObjectBase".

The attribute "instancePathAndName" is available.

"Type" of type "TypeRef", "InitialValue" of type "Value", and "Address" of type "FixedAddressExpression" are explained in 11.5, 11.6, and 11.8.

7 Abstract complex types

7.1 Purpose of abstract complex types

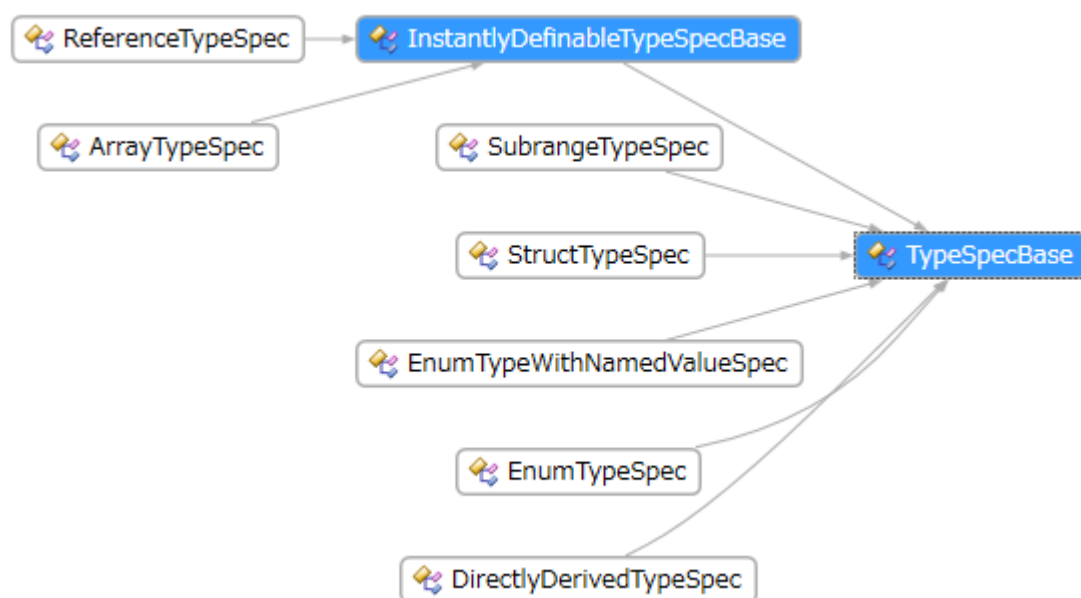
Abstract complex types are provided in the schema for either of the following purposes:

- A) Simply to avoid defining similar attributes and sub-elements in the derived complex type multiple times. For this purpose, common attributes and sub-elements, among derived complex types, are defined in an abstract complex type. This abstract complex type is extended by each related derived complex type.
- B) For vendor-specific extension. For this purpose, the abstract complex type is specified as the type of the element in the standard schema. This allows any concrete complex type which extends the abstract complex type to be applied for the type of the element. In this case, the abstract complex type itself may even contain no attributes nor sub-elements. By adding extended complex type in vendor-specific extension schema, it becomes available as the type for the element.

7.2 Abstract complex types for data type specifications

7.2.1 General

Extension relationships among this kind of abstract complex types and concrete complex types are shown in Figure 14.



IEC

Figure 14 – Extension relationship among complex types for data type specifications

7.2.2 "TypeSpecBase"

The abstract complex type "TypeSpecBase" is provided for vendor-specific extension of user defined data types. It shall be specified as the extension base of the complex type which represents user defined data type specification. It has no specific contents.

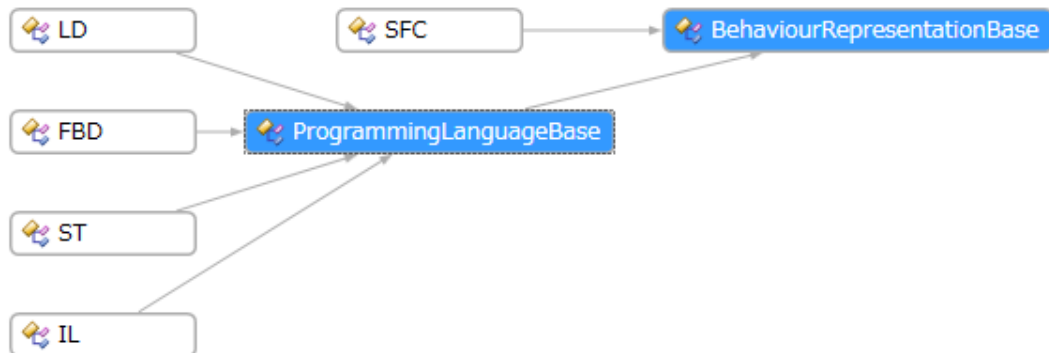
7.2.3 "InstantlyDefinableTypeSpecBase"

The abstract complex type "InstantlyDefinableTypeSpecBase" is also provided for vendor-specific extension of user-defined data types. It shall be specified as the extension base of the complex type which represents user defined data type specification which can be instantly defined within variable declarations. It is based on the abstract complex type "TypeSpecBase" which is explained in 7.2.2, but has no specific contents.

7.3 Abstract complex types for behaviour representations

7.3.1 General

Extension relationships among this kind of abstract complex types and concrete complex types are shown in Figure 15.



IEC

Figure 15 – Extension relationship among complex types for behaviour representations

7.3.2 "BehaviourRepresentationBase"

The abstract complex type "BehaviourRepresentationBase" is provided for vendor-specific extension of behaviour description. It shall be specified as the extension base of the complex type which represents high-level behaviour representation which needs to have some internal states such as SFC. It has no content. Behaviour representations derived from this type is not available within the function nor the method.

7.3.3 "ProgrammingLanguageBase"

The abstract complex type "ProgrammingLanguageBase" is also provided for vendor-specific extension of behaviour description. It shall be specified as the extension base of the complex type which represents a graphical or textual programming language, such as FBD, LD, IL, or ST. It extends "BehaviourRepresentationBase", which is explained in 7.3.2, but has no additional contents. "ProgrammingLanguageBase" is available for function and method.

7.4 Abstract complex types for graphical objects

7.4.1 General

Extension relationships among this kind of abstract complex types and concrete complex types are shown in Figure 16.

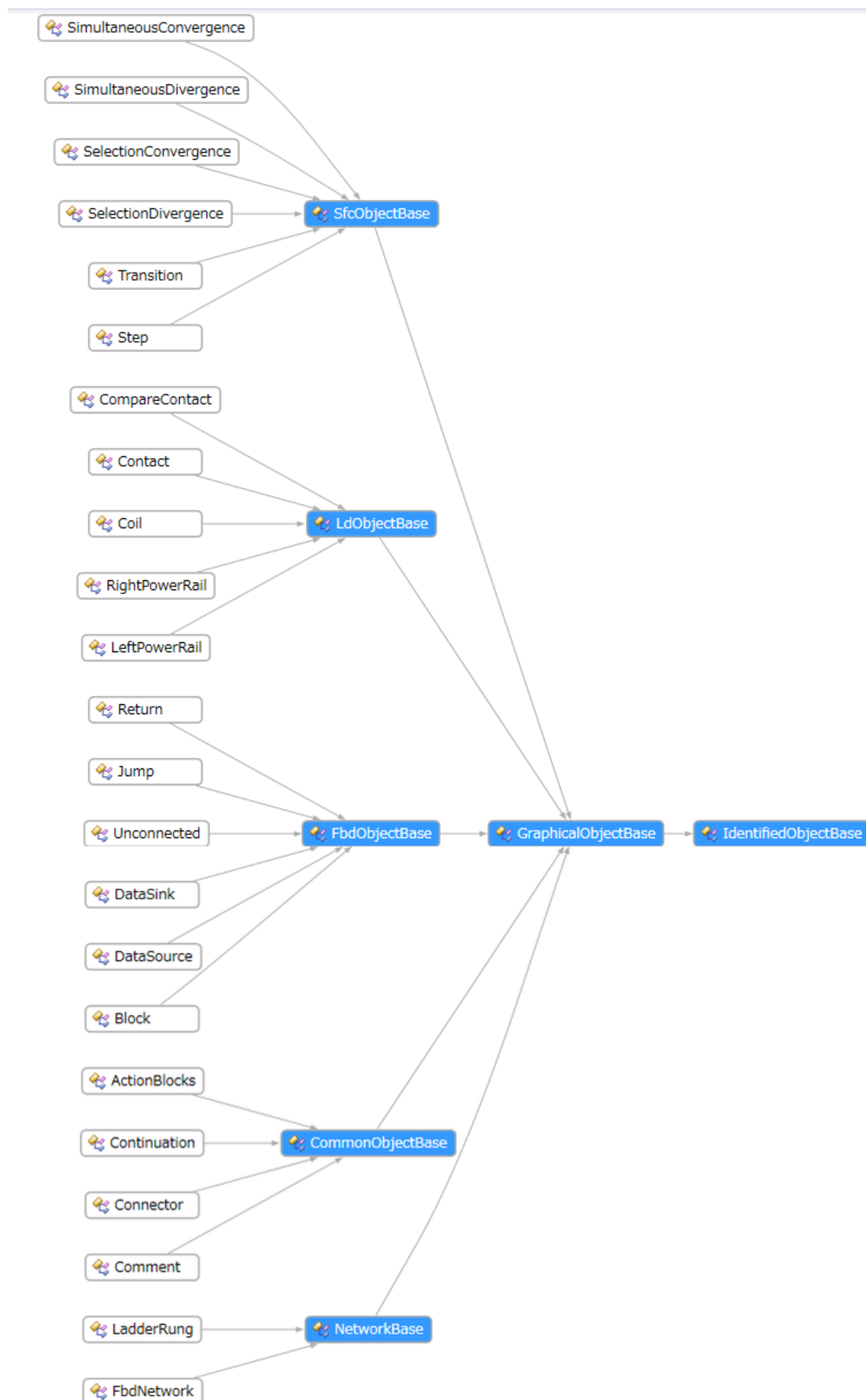


Figure 16 – Extension relationship among complex types for graphical objects

IEC

7.4.2 "IdentifiedObjectBase"

The complex type "IdentifiedObjectBase" is an abstract complex type which contains common data for objects to be identified with "globalId" attribute. Its content is shown in Figure 17.

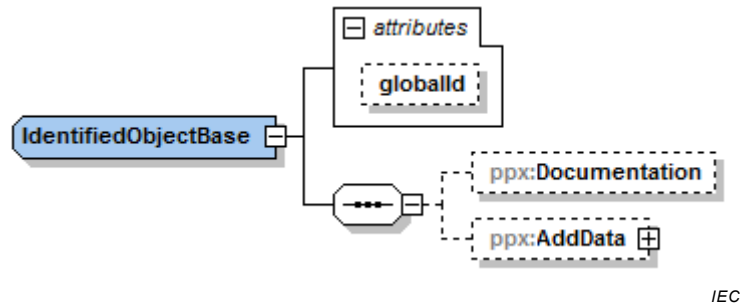


Figure 17 – Complex type "IdentifiedObjectBase"

The main purpose of the attribute "globalId" is the capability to unambiguously identify an element within an XML document from the outside.

The element "Documentation" of complex type "TextBase" represents optional textual documentation for the object. "TextBase" is explained in 15.3.

7.4.3 "GraphicalObjectBase"

The complex type "GraphicalObjectBase" is an abstract complex type which contains common data for graphical objects such as relative position and size. Its content is shown in Figure 18.

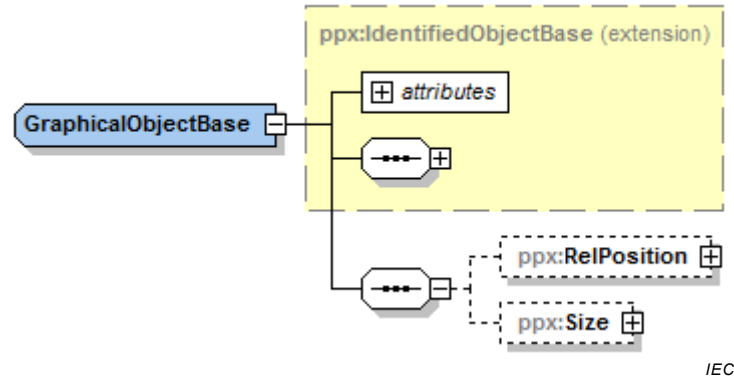


Figure 18 – Complex type "GraphicalObjectBase"

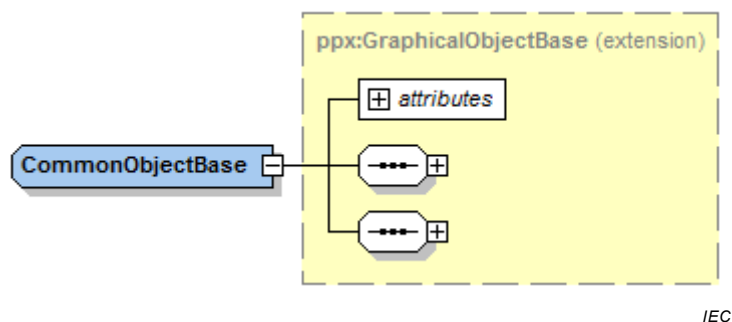
"GraphicalObjectBase" is based on the abstract complex type "IdentifiedObjectBase", which is explained in 7.4.2.

The element "RelPosition" of type complex type "XyDecimalValue", which is explained in 15.1, represents the relative position of the anchor point of the graphical object within the parent graphical object.

The element "Size" of type complex type "XyDecimalValue", which is explained in 15.1, represents the size of the graphical object.

7.4.4 "CommonObjectBase"

The complex type "CommonObjectBase" is an abstract complex type to be the extension base of complex types which represent graphical objects commonly available in FBD, LD and SFC. This complex type may also be used for vendor-specific extension of graphical objects. By adding complex type extended from this abstract complex type, the vendor can make their specific graphical object commonly available in FBD, LD and SFC, in addition to the standard graphical objects such as "Connector", "Continuation", etc. Its content is shown in Figure 19.



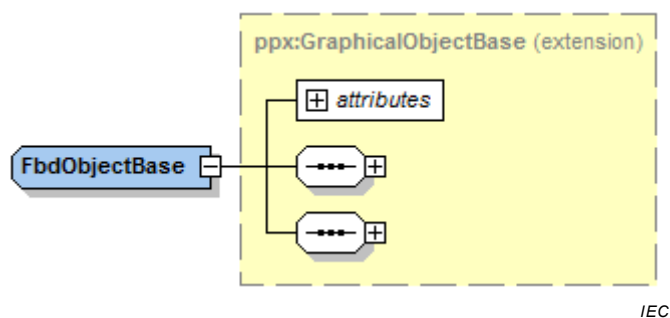
IEC

Figure 19 – Complex type "CommonObjectBase"

This complex type extends the content of the base complex type "GraphicalObjectBase" which is explained in 7.4.3, but itself has no specific contents.

7.4.5 "FbdObjectBase"

The complex type "FbdObjectBase" is an abstract complex type to be the extension base of complex types which represent graphical objects commonly available in FBD and LD. This complex type may also be used for vendor-specific extension of graphical objects. By adding complex type extended from this abstract complex type, the vendor can make their specific graphical object commonly available in FBD and LD in addition to the standard graphical objects such as "Block", "DataSource", "DataSink", etc. Its content is shown in Figure 20.



IEC

Figure 20 – Complex type "FbdObjectBase"

This complex type extends the content of the base complex type "GraphicalObjectBase" which is explained in 7.4.3, but itself has no specific contents.

7.4.6 "LdObjectBase"

The complex type "LdObjectBase" is an abstract complex type to be the extension base of complex types that represent graphical objects only available in LD. This complex type may also be used for vendor-specific extension of graphical objects. By adding complex type extended from this abstract complex type, the vendor can make their specific graphical object only available in LD in addition to the standard graphical objects such as "Coil", "Contact", etc. Its content is shown in Figure 21.

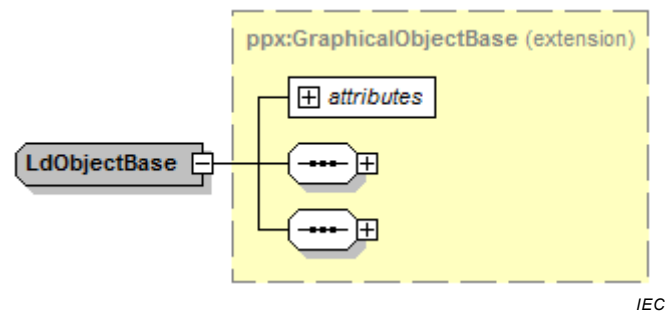


Figure 21 – Complex type "LdObjectBase"

This complex type extends the content of the base complex type "GraphicalObjectBase" which is explained in 7.4.3, but itself has no specific contents.

7.4.7 "SfcObjectBase"

The complex type "SfcObjectBase" is an abstract complex type to be the extension base of complex types which represent graphical objects only available in SFC. It may also be used for vendor-specific extension of graphical objects. By adding complex type extended from this abstract complex type, the vendor can make their specific graphical object only available in SFC in addition to the standard graphical objects such as "Step", "Transition", etc. Its content is shown in Figure 22.

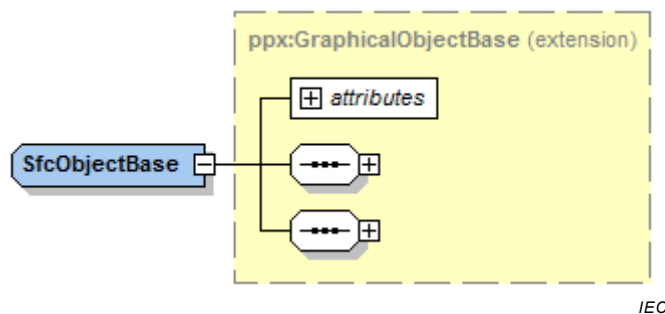
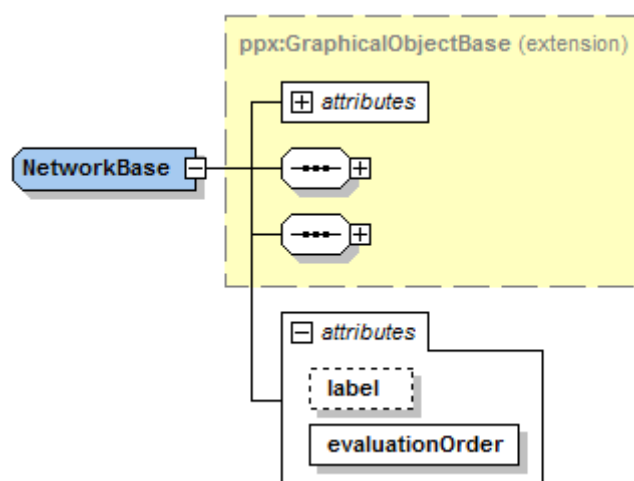


Figure 22 – Complex type "SfcObjectBase"

This complex type extends the content of the base complex type "GraphicalObjectBase" which is explained in 7.4.3, but itself has no specific contents.

7.4.8 "NetworkBase"

The complex type "NetworkBase" is an abstract complex type which contains common attributes and sub-element for the network of LD and FBD. It is specified as the extension base of the complex type of "Rung" element of the complex type "LD", and "FbdNetwork" element of the complex type "FBD". Its content is shown in Figure 23.



IEC

Figure 23 – Complex type "NetworkBase"

"NetworkBase" extends the content of the base complex type "GraphicalObjectBase" which is explained in 7.4.3.

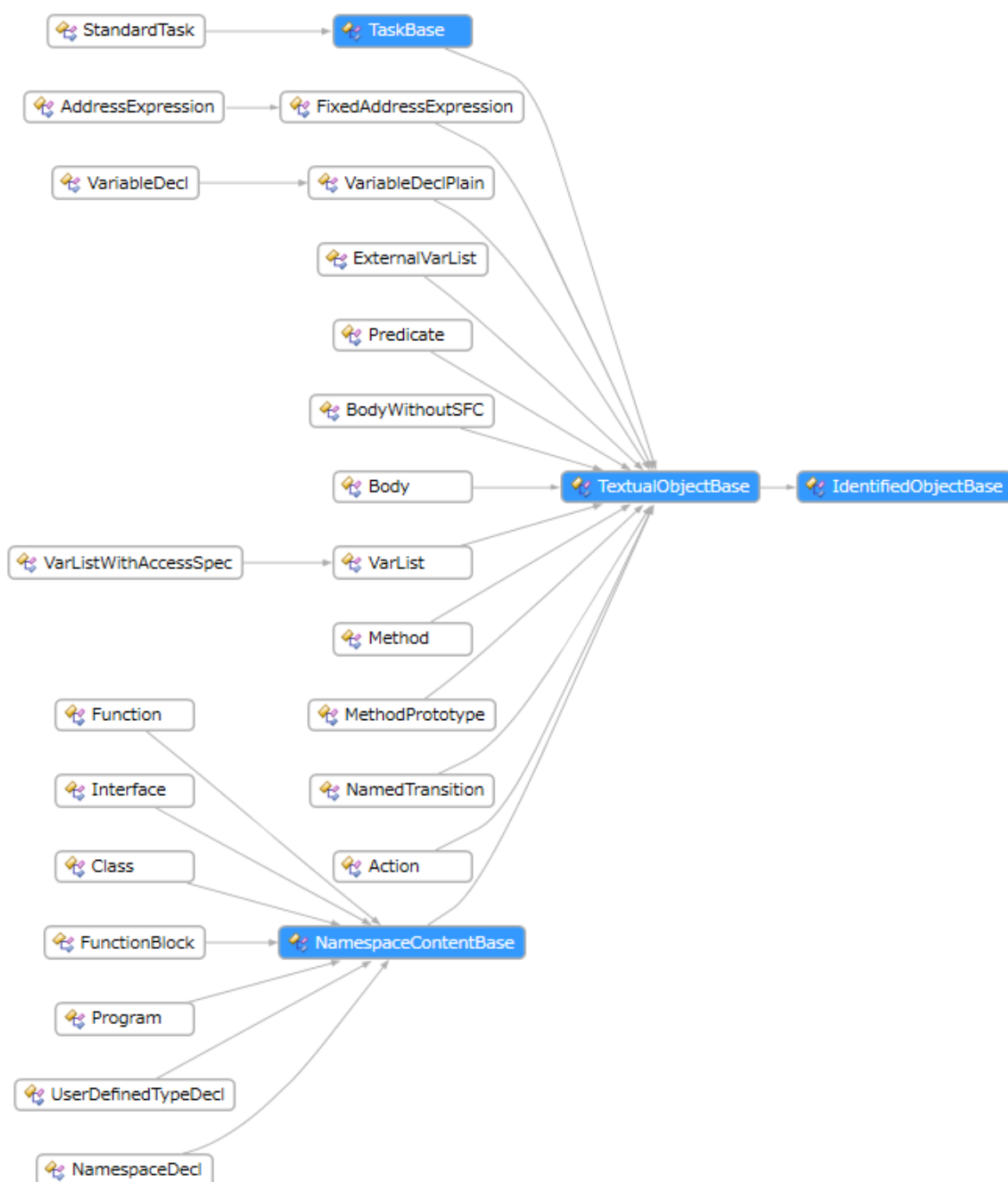
The optional attribute "label" represents the label of the network which can be specified as the target of the execution control element "jump".

The required attribute "evaluationOrder" represents evaluation order of each network within LD or FBD, which have to be significant from the view point of IEC 61131-3. The network with a smaller number of "evaluationOrder" shall be evaluated earlier. It shall not have negative value. This attribute is provided because XML 1.0 specification of W3C does not ensure that appearance order of sibling elements is significant. It is strongly recommended that the exporter should give ascending incremental number to the attribute of each "Rung" element (LD), and each "FbdNetwork" element (LD, FBD). In case of SFC, every "GraphicalExpression" shall have an "evaluationOrder" of "0".

7.5 Abstract complex types for textual constructs

7.5.1 General

Extension relationships among this kind of abstract complex types and concrete complex types are shown in Figure 24.



IEC

Figure 24 – Extension relationship among complex types for textual objects

7.5.2 "TextualObjectBase"

The complex type "TextualObjectBase" is an abstract complex type which contains common data for all textual objects declared by some textual construct. Its content is shown in Figure 25.

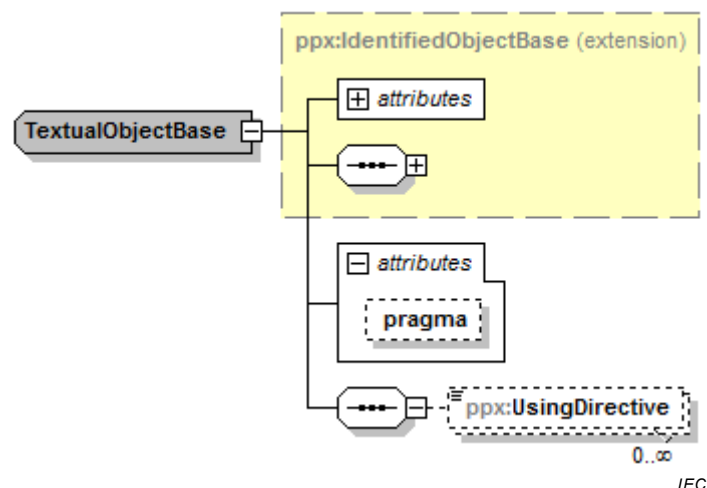


Figure 25 – Complex type "TextualObjectBase"

"TextualObjectBase" is based on the abstract complex type "IdentifiedObjectBase" which is explained in 7.4.2.

The attribute "pragma" is of type string. In order to hold several pragmas within the same string each pragma shall be embedded in curly brackets, as the syntax is defined in IEC 61131-3.

The element "UsingDirective" of type string represents using namespace directive with "USING" keyword, as defined in IEC 61131-3.

7.5.3 "NamespaceContentBase"

The complex type "NamespaceContentBase" is an abstract complex type which contains common data for the contents of namespace such as data types and POU. Its content is shown in Figure 26.

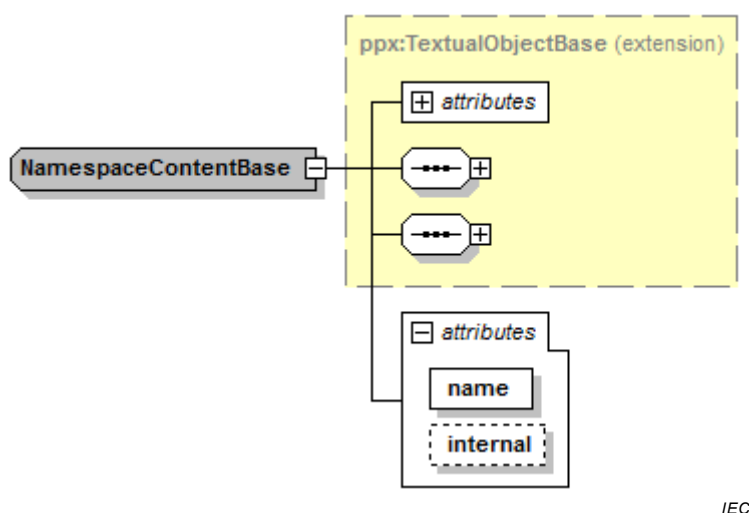


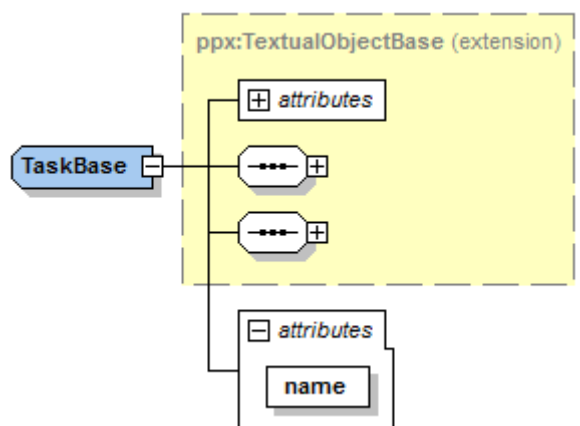
Figure 26 – Complex type "NamespaceContentBase"

"NamespaceContentBase" is based on the abstract complex type "TextualObjectBase", which is explained in 7.5.2.

The required attribute "name" represents the name of the object, such as user-defined data type or POU definition. The optional attribute "internal" represents internal access specifier with keyword "INTERNAL", as defined in IEC 61131-3.

7.5.4 "TaskBase"

The complex type "TaskBase" is an abstract complex type to be the extension base of complex types which represent task as defined in IEC 61131-3. It is provided for vendor-specific extension of task. By adding complex type extending this abstract complex type, the vendor can make their specific task available in addition to the "StandardTask" which also extends this abstract complex type. Its content is shown in Figure 27.



IEC

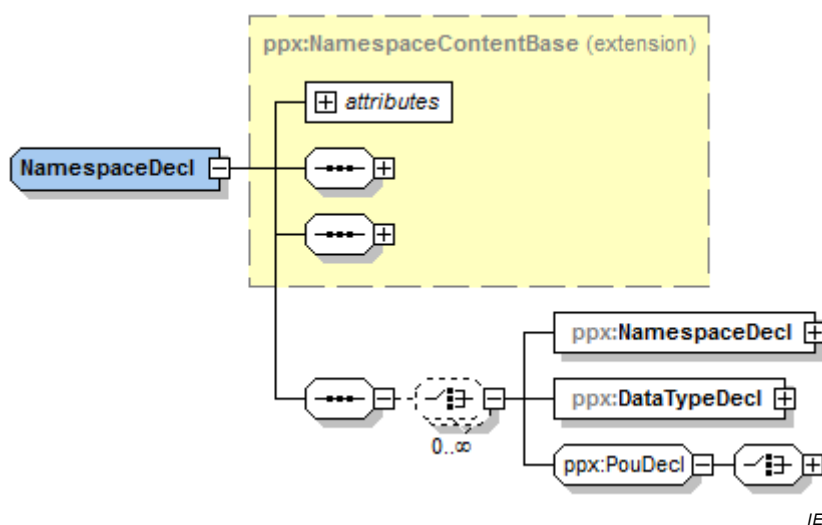
Figure 27 – Complex type "TaskBase"

"TaskBase" is based on the abstract complex type "TextualObjectBase", which is explained in 7.5.2.

The required attribute "name" represents name of the task, which can be referenced by program instance declaration, as defined in IEC 61131-3.

8 Namespace declaration

The complex type "NamespaceDecl" represents namespace, as defined in IEC 61131-3. Its content is shown in Figure 28.



IEC

Figure 28 – Complex type "NamespaceDecl"

A "NamespaceDecl" is based on the abstract complex type "NamespaceContentBase", which is explained in 7.5.3.

Namespaces may be defined recursively, so the element "NamespaceDecl" can be contained in itself.

"DataTypeDecl" of type "UserDefinedTypeDecl" is explained in 9.1. The contents of the group "PouDecl" is explained in 10.1.

9 User-defined data type declaration

9.1 "UserDefinedTypeDecl"

The complex type "UserDefinedTypeDecl" represents all user-defined data-type-related elements, as defined in IEC 61131-3. Its content is shown in Figure 29.

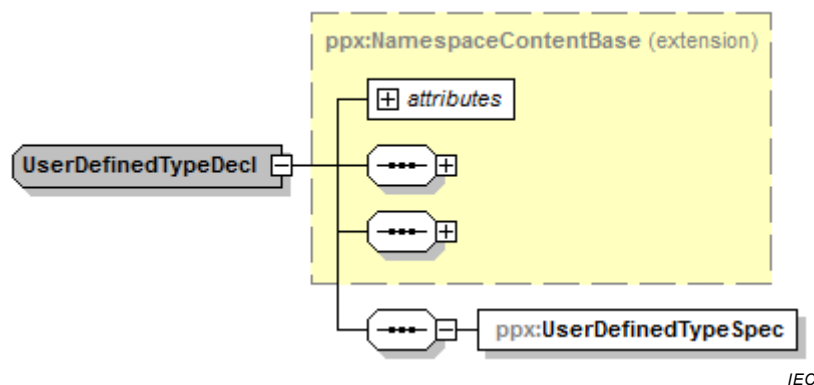


Figure 29 – Complex type "UserDefinedTypeDecl"

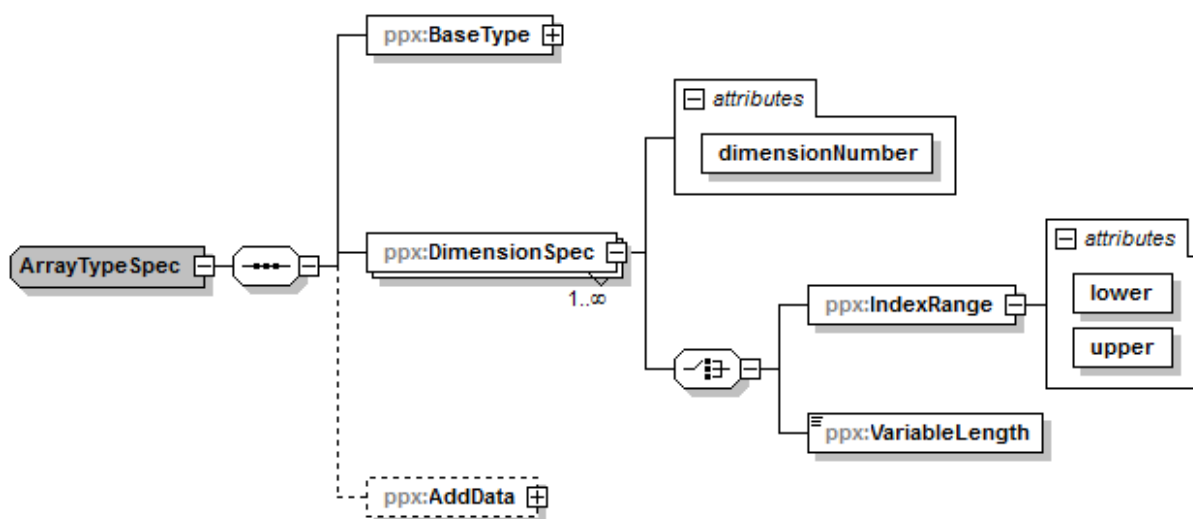
A "UserDefinedTypeDecl" inherits all elements of the type "NamespaceContentBase", which is explained in 7.5.3.

"UserDefinedTypeSpec" of abstract complex type "TypeSpecBase" is explained in 7.2.2. This element shall contain a derived concrete user-defined data type. Subclause 4.4 explains how abstract complex types are to be used.

The possible derived types, as defined in IEC 61131-3, are explained in 9.2, 9.3, 9.4, 9.5, 9.6, 9.7, 9.8, and 9.9.

9.2 "ArrayTypeSpec"

The complex type "ArrayTypeSpec" represents array type, as defined in IEC 61131-3. Its content is shown in Figure 30.



IEC

Figure 30 – Complex type "ArrayTypeSpec"

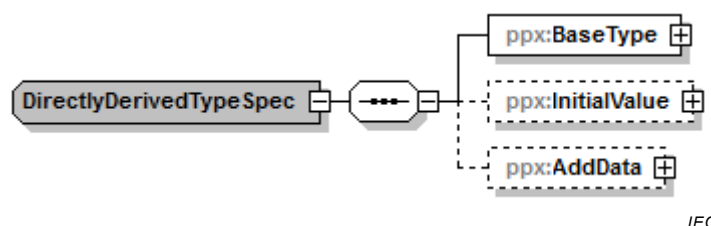
"BaseType" represents the base type name of the array. "BaseType" is of type "TypeRef", which is explained in 11.5.

The "DimensionSpec" shall be used to specify one or more dimensions for the array type. The actual number of dimensions is set as value of the attribute "dimensionNumber".

The "IndexRange" element shall be used to specify the lower and upper border of the dimension as attribute values. If it is a variable length array as defined in IEC 61131-3, then the "VariableLength" element shall be used instead.

9.3 "DirectlyDerivedTypeSpec"

The complex type "DirectlyDerivedTypeSpec" represents directly derived data types, as defined in IEC 61131-3. Its content is shown in Figure 31.



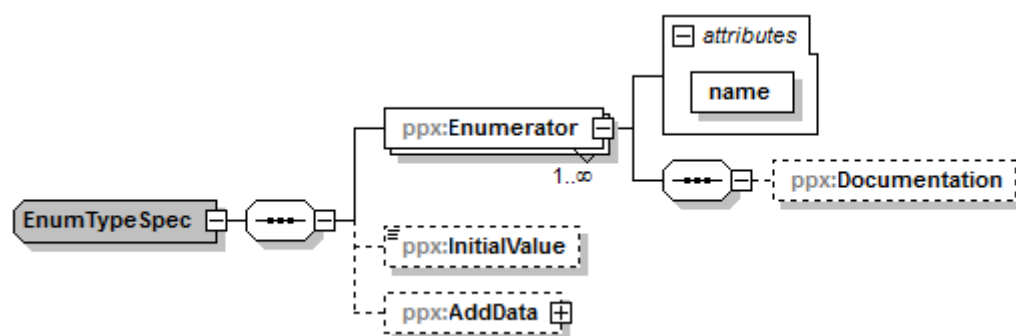
IEC

Figure 31 – Complex type "DirectlyDerivedTypeSpec"

"BaseType" of type "TypeRef" and "InitialValue" of type "Value" are explained in 11.5 and 11.6.

9.4 "EnumTypeSpec"

The complex type "EnumTypeSpec" represents enumerated type, as defined in IEC 61131-3. Its content is shown in Figure 32.



IEC

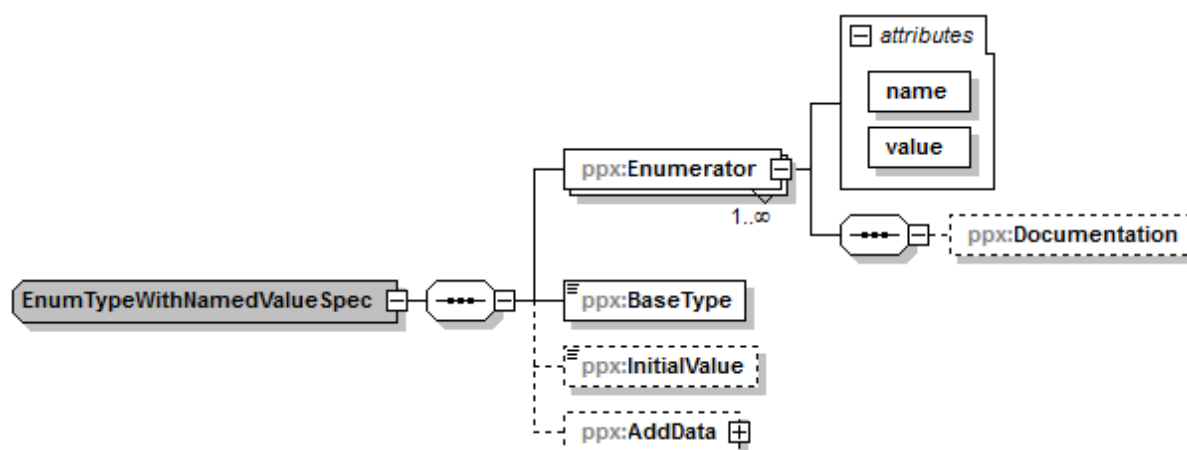
Figure 32 – Complex type "EnumTypeSpec"

For each element of the enumeration list, an element "Enumerator" is required. The attribute "name" stores the associated identifier. "Documentation" of type "TextBase" is explained in 15.3.

"InitialValue" of type string represents the initial value of the enumerated type, which should be the name of one of the enumerators of this type.

9.5 "EnumTypeWithNamedValueSpec"

The complex type "EnumTypeWithNamedValueSpec" represents data type with named values, as defined in IEC 61131-3. Its content is shown in Figure 33.



IEC

Figure 33 – Complex type "EnumTypeWithNamedValueSpec"

For each element of the enumeration list an element "Enumerator" is required. The attribute "name" stores the associated identifier. Additionally, there is an attribute "value" for storage of the enumeration value associated with the identifier. "Documentation" of type "TextBase" is explained in 15.3.

"BaseType" is of type "ElementaryType", which is explained in 9.9. "InitialValue" is of type string.

9.6 "StructTypeSpec"

The complex type "StructTypeSpec" represents structured type, as defined in IEC 61131-3. Its content is shown in Figure 34.

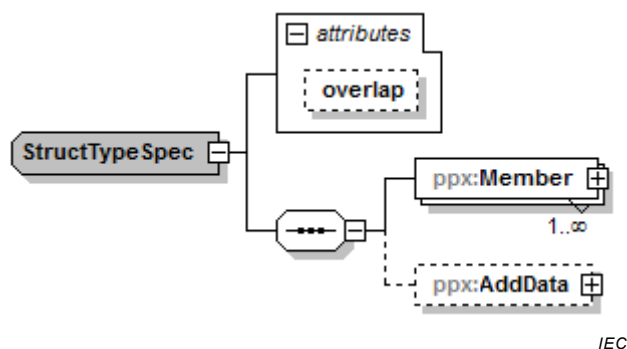


Figure 34 – Complex type "StructTypeSpec"

The attribute "overlap" maps to the OVERLAP keyword in structured data type declarations. "Member" of type "VariableDecl" is explained in 11.3.

9.7 "SubrangeTypeSpec"

The complex type "SubrangeTypeSpec" represents sub range data types, as defined in IEC 61131-3. Its content is shown in Figure 35.

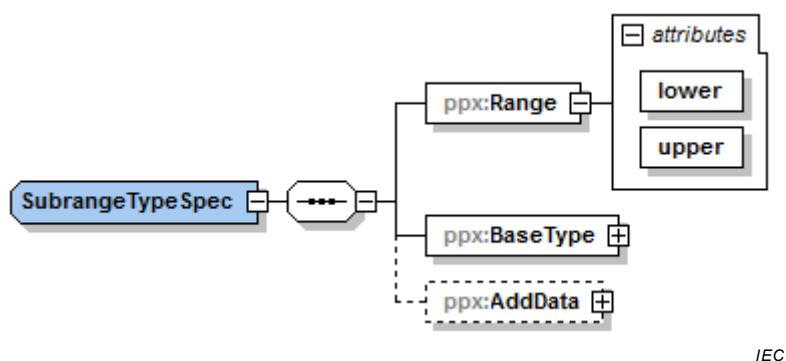


Figure 35 – Complex type "SubrangeTypeSpec"

The limits of the sub range type shall be given in the attributes "lower" and "upper" of the element "Range".

"BaseType" of type "TypeRef" is explained in 11.5.

9.8 "ReferenceTypeSpec"

The complex type "ReferenceTypeSpec" represents references, as defined in IEC 61131-3. Its content is shown in Figure 36.

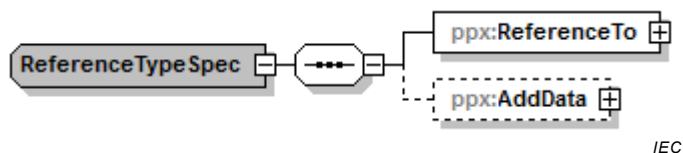


Figure 36 – Complex type "ReferenceTypeSpec"

"ReferenceTo" of type "TypeRef" is explained in 11.5.

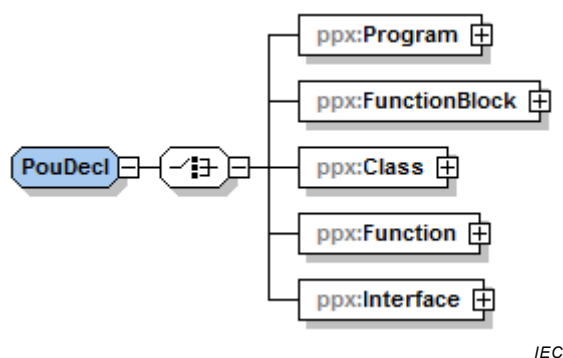
9.9 "ElementaryType"

The simple type "ElementaryType" represents all elementary data types, as defined in IEC 61131-3. It is of type string and the possible values are restricted to the names used in IEC 61131-3 for the elementary data types.

10 POU declaration

10.1 "PouDecl"

The group "PouDecl" represents a choice between the different POUs (program organisation units), as defined in IEC 61131-3. Its content is shown in Figure 37.



IEC

Figure 37 – Complex type "PouDecl"

"Program" of type "Program", "FunctionBlock" of type "FunctionBlock", "Class" of type "Class", "Function" of type "Function", and "Interface" of type "Interface" are explained in 10.2, 10.3, 10.4, 10.5, and 10.5.

10.2 "Program"

The complex type "Program" represents a program, as defined in IEC 61131-3. Its content is shown in Figure 38.

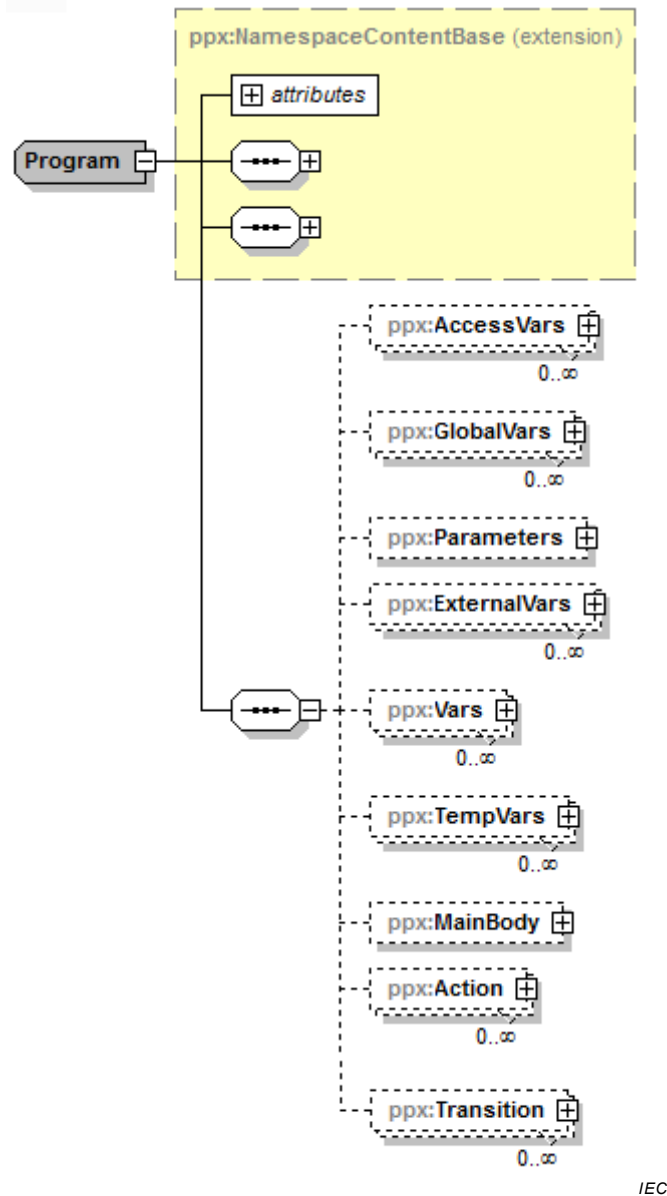


Figure 38 – Complex type "Program"

This complex type extends the content of the base complex type "NamespaceContentBase" which is explained in 7.5.3.

"AccessVars" and "GlobalVars" are of type "VarList", which is explained in 11.1.

"Parameters" of type "ParameterSet" is explained in 10.11.

"ExternalVars" of type "ExternalVarList" and "Vars" of type "VarListWithAccessSpec" are explained in 11.2.

"TempVars" of type "VarList" is explained in 11.1.

"MainBody" of type "Body" is explained in 10.14.

"Action" of type "Action" and "Transition" of type "NamedTransition" are explained in 10.7 and 10.8.

10.3 "FunctionBlock"

The complex type "FunctionBlock" represents a function block, as defined in IEC 61131-3. Its content is shown in Figure 39.

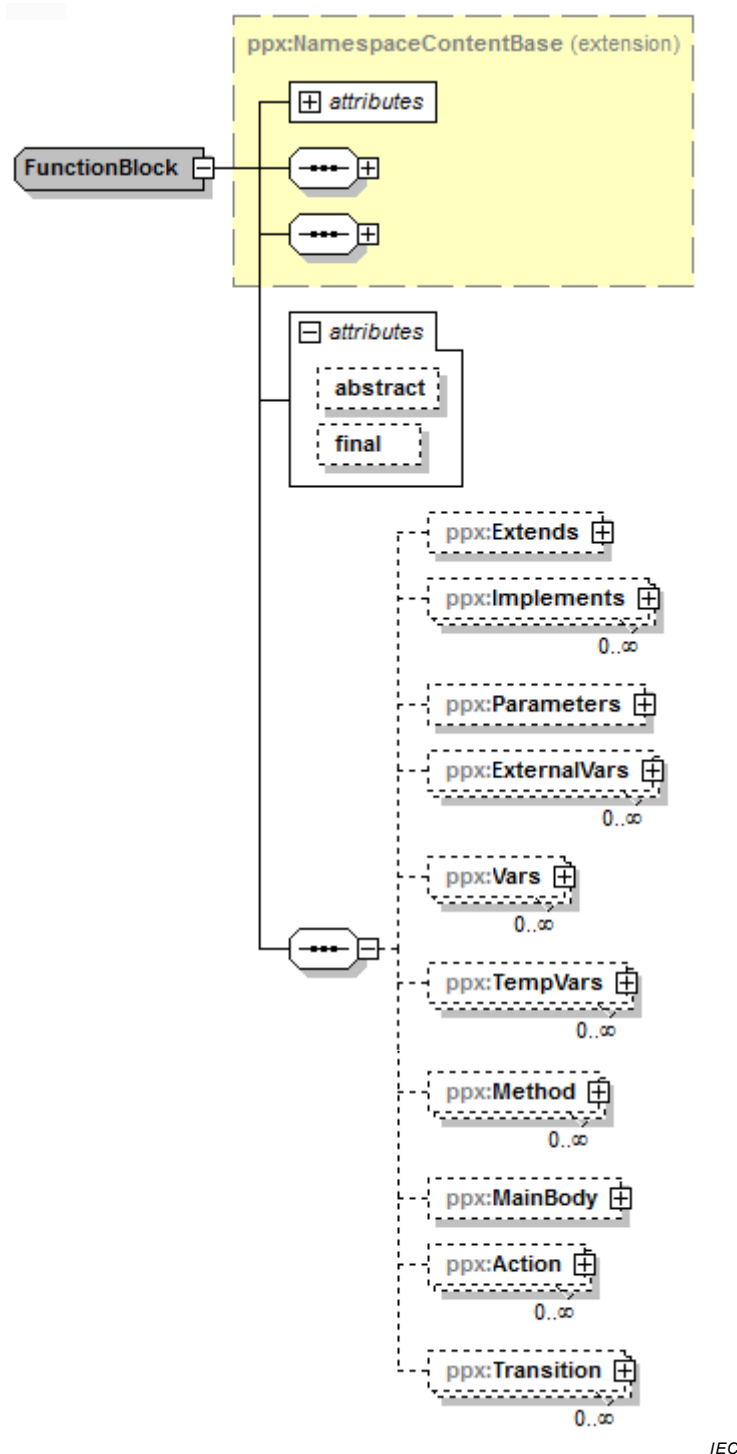


Figure 39 – Complex type "FunctionBlock"

This complex type extends the content of the base complex type "NamespaceContentBase", which is explained in 7.5.3.

The attributes "abstract" and "final" represent whether the function block has been defined using the keywords ABSTRACT and FINAL, as defined in IEC 61131-3.

"Extends" contains the name of base class or function block if applicable. "Implements" contains the implemented interfaces if applicable. Both are of type "typeRef" which is explained in 11.5.

"Parameters" of type "ParameterSet" is explained in 10.11.

"ExternalVars" of type "ExternalVarList" is explained in 11.2.

"Vars" of type "VarListWithAccessSpec" is explained in 10.12.

"TempVars" of type "VarList" is explained in 11.1.

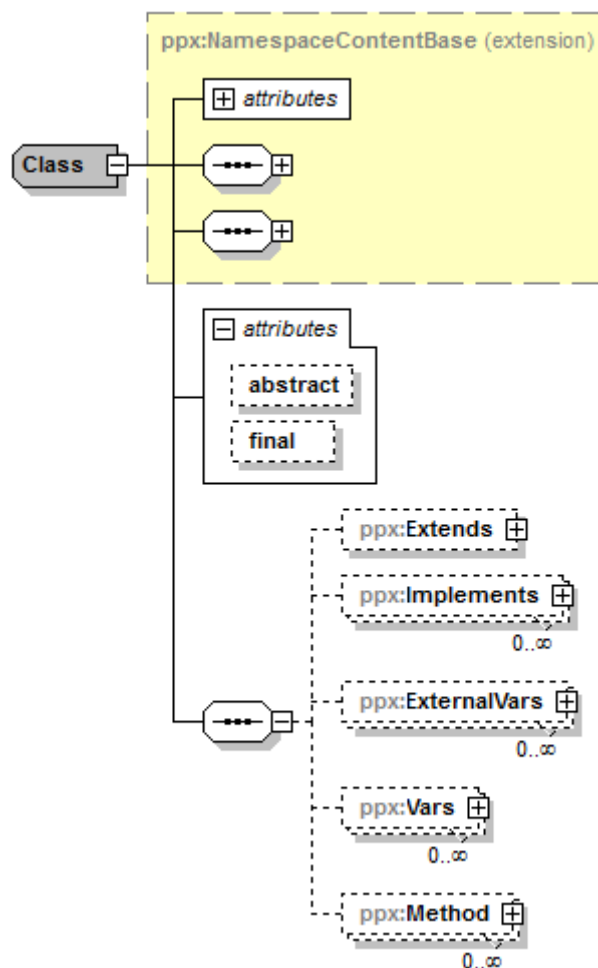
"Method" of type "Method" is explained in 10.10.

"MainBody" of type "Body" is explained in 10.14.

"Action" of type "Action" and "Transition" of type "NamedTransition" are explained in 10.7 and 10.8.

10.4 "Class"

The complex type "Class" represents a class, as defined in IEC 61131-3. Its content is shown in Figure 40.



IEC

Figure 40 – Complex type "Class"

This complex type extends the content of the base complex type "NamespaceContentBase" which is explained in 7.5.3.

The attributes "abstract" and "final" represent whether the class has been defined using the keywords ABSTRACT and FINAL, as defined in IEC 61131-3.

"Extends" contains the name of base class if applicable. "Implements" contains the implemented interfaces if applicable. Both are of type "typeRef" which is explained in 11.5.

"ExternalVars" of type "ExternalVarList" is explained in 11.2.

"Vars" of type "VarListWithAccessSpec" is explained in 10.12.

"Method" of type "Method" is explained in 10.10.

10.5 "Function"

The complex type "Function" represents a function, as defined in IEC 61131-3. Its content is shown in Figure 41.

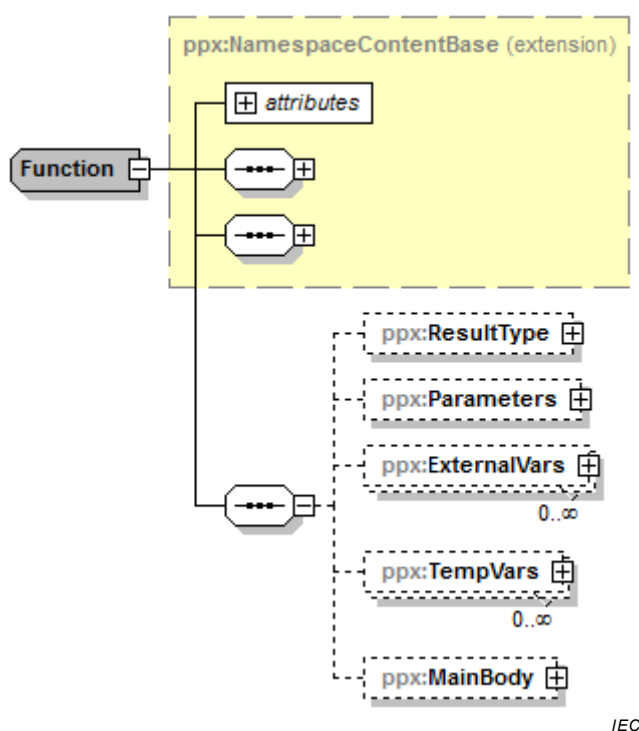


Figure 41 – Complex type "Function"

This complex type extends the content of the base complex type "NamespaceContentBase" which is explained in 7.5.3.

"ResultType" contains the name of the data type being delivered by the function. "ResultType" of type "TypeRef" is explained in 11.5.

"Parameters" of type "ParameterSet" is explained in 10.11.

"ExternalVars" of type "ExternalVarList" is explained in 11.2.

"TempVars" of type "VarList" is explained in 11.1.