

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Function Blocks (FB) for process control and Electronic Device Description
Language (EDDL) –
Part 3: EDDL syntax and semantics**

**Blocs Fonctionnels (FB) pour les procédés industriels et le Langage de
Description Electronique de Produit (EDDL) –
Partie 3: Sémantique et syntaxe EDDL**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2015 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC

La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC

Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue

Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub

La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,...). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 15 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

Plus de 60 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Function Blocks (FB) for process control and Electronic Device Description
Language (EDDL) –
Part 3: EDDL syntax and semantics**

**Blocs Fonctionnels (FB) pour les procédés industriels et le Langage de
Description Electronique de Produit (EDDL) –
Partie 3: Sémantique et syntaxe EDDL**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 25.040.40; 35.240.50

ISBN 978-2-8322-2708-4

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	16
INTRODUCTION.....	18
1 Scope.....	19
2 Normative references.....	19
3 Terms, definitions, abbreviated terms and acronyms	20
3.1 Terms and definitions	20
3.2 Abbreviated terms and acronyms	22
4 Conformance statement	22
5 Conventions	23
5.1 General.....	23
5.2 Conventions for lexical structure.....	23
5.2.1 ABC field1, field2	23
5.2.2 ABC field1+	24
5.2.3 ABC field2*	24
5.2.4 ABC [field1, field2]+	24
5.2.5 ABC field1, (field2, field3)<exp>	24
6 EDD and EDDL model	24
6.1 Overview of EDD and EDDL.....	24
6.2 EDD architecture.....	24
6.3 Concepts of EDD	24
6.4 Principles of the EDD development process.....	25
6.4.1 General	25
6.4.2 EDD source generation	25
6.4.3 EDD preprocessing	26
6.4.4 EDD compilation	26
6.5 Interrelations between the lexical structure and formal definitions.....	26
6.6 Builtins	26
6.7 Profiles	26
7 Electronic Device Description Language (EDDL)	26
7.1 Overview.....	26
7.1.1 EDDL features	26
7.1.2 Syntax representation	27
7.1.3 EDD language elements.....	27
7.1.4 Basic construction elements	27
7.1.5 Common attributes.....	38
7.1.6 Special elements.....	38
7.1.7 Rules for instances	39
7.1.8 Rules for a list of VARIABLES.....	39
7.2 EDD identification information	39
7.2.1 General structure	39
7.2.2 Specific attributes	39
7.3 AXIS.....	42
7.3.1 General structure	42
7.3.2 Specific attributes	43
7.4 BLOCK	45
7.4.1 BLOCK_A	45

7.4.2	BLOCK_B	57
7.5	CHART	59
7.5.1	General structure	59
7.5.2	Specific attributes	59
7.6	COLLECTION	61
7.6.1	General structure	61
7.6.2	Specific attribute – item-type	62
7.7	COMMAND	62
7.7.1	General structure	62
7.7.2	Specific attributes	63
7.8	COMPONENT	69
7.8.1	General structure	69
7.8.2	Specific attributes	70
7.9	COMPONENT_FOLDER	75
7.10	COMPONENT_REFERENCE	76
7.11	COMPONENT_RELATION	76
7.11.1	General structure	76
7.11.2	Specific attributes	77
7.12	CONNECTION (void)	80
7.13	DOMAIN (void)	80
7.14	EDIT_DISPLAY	80
7.14.1	General structure	80
7.14.2	Specific attributes	81
7.15	FILE	83
7.15.1	General structure	83
7.15.2	Specific attributes	83
7.16	GRAPH	84
7.16.1	General structure	84
7.16.2	Specific attributes	85
7.17	GRID	86
7.17.1	General structure	86
7.17.2	Specific attributes	86
7.18	IMAGE	87
7.18.1	General structure	87
7.18.2	Specific attributes	88
7.19	IMPORT	89
7.19.1	General structure	89
7.19.2	Redefinitions	91
7.20	INTERFACE	107
7.20.1	General structure	107
7.20.2	Specific attribute – DECLARATION	107
7.21	LIKE	108
7.22	LIST	108
7.22.1	General structure	108
7.22.2	Specific attributes	109
7.23	MENU	110
7.23.1	General structure	110
7.23.2	Specific attributes	111
7.23.3	Sequence diagrams for actions	116

7.24	METHOD	118
7.24.1	General structure	118
7.24.2	Specific attributes	119
7.25	PROGRAM (void)	121
7.26	RECORD	121
7.27	REFERENCE_ARRAY	122
7.27.1	General structure	122
7.27.2	Specific attribute – ELEMENTS	122
7.28	Relations	123
7.28.1	REFRESH	123
7.28.2	UNIT	123
7.28.3	WRITE_AS_ONE	124
7.29	RESPONSE_CODES	124
7.30	SOURCE	125
7.30.1	General structure	125
7.30.2	Specific attributes	126
7.31	TEMPLATE	127
7.31.1	General structure	127
7.31.2	Specific attribute – DEFAULT_VALUES	127
7.32	VALUE_ARRAY	128
7.32.1	General structure	128
7.32.2	Specific attributes	128
7.33	VARIABLE	129
7.33.1	General structure	129
7.33.2	Specific attributes	130
7.34	VARIABLE_LIST	148
7.35	WAVEFORM	148
7.35.1	General structure	148
7.35.2	Specific attributes	149
7.36	Common attributes	155
7.36.1	CLASSIFICATION	155
7.36.2	COMPONENT_PARENT	156
7.36.3	COMPONENT_PATH	157
7.36.4	DEFINITION	157
7.36.5	EMPHASIS	158
7.36.6	HANDLING	158
7.36.7	HEIGHT	158
7.36.8	HELP	159
7.36.9	LABEL	160
7.36.10	LINE_COLOR	160
7.36.11	LINE_TYPE	160
7.36.12	MEMBERS	161
7.36.13	PROTOCOL	162
7.36.14	RESPONSE_CODES	163
7.36.15	SUPPLIED_INTERFACE	163
7.36.16	VALIDITY	163
7.36.17	WIDTH	164
7.36.18	PRIVATE	164
7.36.19	VISIBILITY	164

7.36.20	WRITE_MODE	165
7.36.21	IDENTITY	165
7.37	Conditional expression	166
7.38	Referencing	167
7.38.1	Referencing an EDD instance	167
7.38.2	Referencing bits of a BIT_ENUMERATED VARIABLE	168
7.38.3	Referencing members of a RECORD	168
7.38.4	Referencing elements of a VALUE_ARRAY	168
7.38.5	Referencing members of a COLLECTION	169
7.38.6	Referencing elements of a REFERENCE_ARRAY	169
7.38.7	Referencing members of a VARIABLE_LISTS	169
7.38.8	Referencing elements of BLOCK_A PARAMETERS	170
7.38.9	Referencing elements of BLOCK_A PARAMETER_LISTS	170
7.38.10	Referencing elements of BLOCK_A LOCAL_PARAMETERS	170
7.38.11	Referencing BLOCK_A CHARACTERISTICS	171
7.38.12	Referencing members of a FILE	171
7.38.13	Referencing elements of a LIST	171
7.38.14	Referencing members of a CHART	172
7.38.15	Referencing members of a GRAPH	172
7.38.16	Referencing members of a SOURCE	172
7.38.17	Referencing AXIS of a GRAPH, SOURCE, WAVEFORM	173
7.38.18	Referencing PARAMETERS of specific BLOCK_A instance	173
7.38.19	Referencing LOCAL_PARAMETERS of specific BLOCK_A instance	174
7.38.20	Referencing CHARACTERISTICS of specific BLOCK_A instance	174
7.38.21	Referencing CHARTS of specific BLOCK_A instance	174
7.38.22	Referencing LISTS of specific BLOCK_A instance	175
7.38.23	Referencing GRAPHS of specific BLOCK_A instance	175
7.38.24	Referencing GRIDS of specific BLOCK_A instance	176
7.38.25	Referencing MENUS of specific BLOCK_A instance	176
7.38.26	Referencing METHODS of specific BLOCK_A instance	177
7.38.27	Referencing COMPONENT instances	177
7.38.28	Referencing COMPONENT types	178
7.38.29	Referencing FILES of specific BLOCK_A instance	178
7.38.30	Referencing PLUGINS of specific BLOCK_A instance	178
7.39	Strings	179
7.39.1	Specifying a string as a string literal	179
7.39.2	Specifying a string as a string variable	179
7.39.3	Specifying a string as an enumeration value	180
7.39.4	Specifying a string as a dictionary reference	180
7.39.5	Referencing HELP and LABEL attributes of EDD instances	180
7.39.6	String operations	181
7.39.7	Prompt string formats	181
7.40	Expression	182
7.40.1	General structure	182
7.40.2	Primary expressions	182
7.40.3	Unary expressions	185
7.40.4	Binary expressions	186
7.41	Text dictionary	188
7.42	PLUGIN	189

7.42.1	General structure	189
7.42.2	Specific attribute – UUID	189
7.43	BLOB	190
Annex A	(normative) EDDL formal definition	191
A.1	EDDL preprocessor	191
A.1.1	General structure	191
A.1.2	Directives	191
A.1.3	Predefined macros	194
A.1.4	NEWLINE characters	195
A.1.5	Comments	195
A.2	Conventions	195
A.2.1	Integer constants	195
A.2.2	Floating-point constants	195
A.2.3	String literals	196
A.2.4	Using language and country codes in string literals	196
A.3	Operators	197
A.4	Keywords	201
A.5	Terminals	205
A.6	Formal EDDL syntax	206
A.6.1	General	206
A.6.2	EDD identification information	206
A.6.3	AXIS	207
A.6.4	BLOCK_A and BLOCK_B	208
A.6.5	CHART	212
A.6.6	COLLECTION	213
A.6.7	COMMAND	214
A.6.8	COMPONENT	217
A.6.9	COMPONENT_FOLDER	220
A.6.10	COMPONENT_REFERENCE	220
A.6.11	COMPONENT_RELATION	221
A.6.12	CONNECTION (void)	223
A.6.13	DOMAIN (void)	223
A.6.14	EDIT_DISPLAY	223
A.6.15	FILE	224
A.6.16	GRAPH	224
A.6.17	GRID	225
A.6.18	IMAGE	226
A.6.19	INTERFACE	227
A.6.20	LIST	227
A.6.21	IMPORT	227
A.6.22	LIKE	229
A.6.23	MENU	231
A.6.24	METHOD	233
A.6.25	PROGRAM (void)	234
A.6.26	RECORD	234
A.6.27	REFERENCE_ARRAY	235
A.6.28	Relations	235
A.6.29	RESPONSE_CODES	237
A.6.30	SOURCE	238

A.6.31	TEMPLATE	238
A.6.32	VALUE_ARRAY	239
A.6.33	VARIABLE	239
A.6.34	VARIABLE_LIST	249
A.6.35	WAVEFORM	249
A.6.36	Common attributes	251
A.6.37	Expression	255
A.6.38	C-Grammar	257
A.6.39	Redefinition	261
A.6.40	References	285
A.6.41	PLUGIN	287
A.6.42	BLOB	288
A.7	Formal dictionary syntax	288
Annex B (normative)	EDDL Builtin library (void)	289
Annex C (informative)	EDD example	290
C.1	EDD example of a temperature transmitter	290
C.2	EDD example	291
Annex D (normative)	Profiles of EDDL and Builtins	304
D.1	Conventions for profiles of EDDL and Builtins	304
D.2	Profiles for PROFIBUS and PROFINET	305
D.2.1	EDDL profile	305
D.2.2	Builtin profile	311
D.2.3	EDDL Formal Definition profile	311
D.3	Profiles for FOUNDATION™ fieldbus	312
D.3.1	EDDL profile	312
D.3.2	Builtin profile	318
D.3.3	EDDL Formal Definition profile	319
D.4	Profiles for HART® Communication Foundation (HCF)	319
D.4.1	EDDL profile	319
D.4.2	Builtin profile	326
D.4.3	EDDL Formal Definition profile	326
D.5	Profiles for Communication Servers	326
D.5.1	EDDL profile	326
D.5.2	Builtin profile	333
D.5.3	EDDL Formal Definition profile	333
D.6	Data types	333
D.6.1	METHOD DEFINITION data types	333
D.6.2	VARIABLE TYPE data types	334
Bibliography		340
Figure 1	– Position of IEC 61804 in relation to other standards and products	18
Figure 2	– EDD generation process	25
Figure 3	– BLOCK_A	28
Figure 4	– CHART	28
Figure 5	– COLLECTION	29
Figure 6	– COMMAND	29
Figure 7	– COMPONENT	30

Figure 8 – COMPONENT_FOLDER.....	30
Figure 9 – COMPONENT_REFERENCE.....	30
Figure 10 – COMPONENT_RELATION.....	31
Figure 11 – EDIT_DISPLAY	31
Figure 12 – FILE.....	31
Figure 13 – GRAPH	32
Figure 14 – GRID.....	32
Figure 15 – IMAGE	32
Figure 16 – LIKE.....	33
Figure 17 – LIST.....	33
Figure 18 – MENU	34
Figure 19 – RECORD.....	34
Figure 20 – REFERENCE_ARRAY	35
Figure 21 – REFRESH	35
Figure 22 – UNIT	36
Figure 23 – WRITE_AS_ONE.....	36
Figure 24 – SOURCE.....	36
Figure 25 – VALUE_ARRAY.....	37
Figure 26 – VARIABLE.....	37
Figure 27 – VARIABLE_LIST.....	37
Figure 28 – WAVEFORM	38
Figure 29 – EDDL import mechanisms.....	89
Figure 30 – MENU activation.....	117
Figure C.1 – Example of an operator screen using EDD.....	290
Table 1 – Field attribute descriptions.....	23
Table 2 – DD_REVISION attribute.....	40
Table 3 – DEVICE_REVISION attribute	40
Table 4 – DEVICE_TYPE attributes.....	41
Table 5 – EDD_PROFILE attribute	41
Table 6 – EDD_VERSION attribute.....	41
Table 7 – MANUFACTURER attributes	42
Table 8 – MANUFACTURER_EXT attribute	42
Table 9 – AXIS attributes	43
Table 10 – MAX_VALUE, MIN_VALUE attributes	44
Table 11 – SCALING attributes	44
Table 12 – BLOCK_A attributes.....	46
Table 13 – CHARACTERISTIC attribute	47
Table 14 – PARAMETER attributes	47
Table 15 – AXIS_ITEMS attribute.....	47
Table 16 – CHART_ITEMS attribute	48
Table 17 – COLLECTION_ITEMS attribute	48
Table 18 – EDIT_DISPLAY_ITEMS attribute.....	48

Table 19 – FILE_ITEMS attribute	49
Table 20 – GRAPH_ITEMS attribute	49
Table 21 – GRID_ITEMS attribute	49
Table 22 – IMAGE_ITEMS attribute	50
Table 23 – LIST_ITEMS attribute	50
Table 24 – MENU_ITEMS attribute	51
Table 25 – METHOD_ITEMS attribute	51
Table 26 – PARAMETER_LISTS attributes	51
Table 27 – REFERENCE_ARRAY_ITEMS attribute	52
Table 28 – REFRESH_ITEMS attribute	52
Table 29 – SOURCE_ITEMS attribute	52
Table 30 – UNIT_ITEMS attribute	53
Table 31 – WAVEFORM_ITEMS attribute	53
Table 32 – WRITE_AS_ONE_ITEMS attribute	53
Table 33 – CHARTS attributes	54
Table 34 – LISTS attributes	54
Table 35 – GRAPHS attributes	54
Table 36 – GRIDS attributes	55
Table 37 – MENUS attributes	55
Table 38 – METHODS attributes	56
Table 39 – FILES attributes	56
Table 40 – PLUGIN_ITEMS attribute	56
Table 41 – PLUGINS attributes	57
Table 42 – BLOCK_B attributes	57
Table 43 – NUMBER attributes	58
Table 44 – TYPE attributes	58
Table 45 – CHART attributes	59
Table 46 – CYCLE_TIME attribute	60
Table 47 – LENGTH attribute	60
Table 48 – TYPE attributes	61
Table 49 – COLLECTION attributes	61
Table 50 – item-type	62
Table 51 – COMMAND attributes	63
Table 52 – OPERATION attributes	63
Table 53 – TRANSACTION attributes	64
Table 54 – REPLY and REQUEST attributes	65
Table 55 – INDEX attributes	66
Table 56 – BLOCK_B attribute	66
Table 57 – NUMBER attribute	67
Table 58 – SLOT attributes	67
Table 59 – SUB_SLOT attributes	67
Table 60 – HEADER attribute	68
Table 61 – API attributes	68

Table 62 – POST_RQSTRECEIVE_ACTIONS attribute	69
Table 63 – COMPONENT attributes	70
Table 64 – CAN_DELETE attributes	70
Table 65 – CHECK_CONFIGURATION attribute	71
Table 66 – COMPONENT_RELATIONS attribute	71
Table 67 – DECLARATION attribute	71
Table 68 – DETECT attribute	72
Table 69 – EDD attribute	72
Table 70 – INITIAL_VALUES attributes	73
Table 71 – REDUNDANCY attribute	73
Table 72 – SCAN attribute	73
Table 73 – SCAN_LIST attribute	74
Table 74 – BYTE_ORDER attributes	74
Table 75 – CONNECTION_POINT attribute	75
Table 76 – PRODUCT_URI attribute	75
Table 77 – COMPONENT_FOLDER attributes	75
Table 78 – COMPONENT_REFERENCE attributes	76
Table 79 – COMPONENT_RELATION attributes	77
Table 80 – COMPONENTS attributes	78
Table 81 – RELATION_TYPE attributes	79
Table 82 – ADDRESSING attribute	79
Table 83 – MAXIMUM_NUMBER attribute	79
Table 84 – MINIMUM_NUMBER attribute	80
Table 85 – REQUIRED_INTERFACE attribute	80
Table 86 – EDIT_DISPLAY attributes	81
Table 87 – EDIT_ITEMS attribute	81
Table 88 – DISPLAY_ITEM attribute	82
Table 89 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS attributes	83
Table 90 – FILE attributes	83
Table 91 – SHARED attributes	84
Table 92 – ON_UPDATE_ACTIONS attribute	84
Table 93 – GRAPH attributes	85
Table 94 – CYCLE_TIME attribute	85
Table 95 – X_AXIS attribute	86
Table 96 – GRID attributes	86
Table 97 – VECTORS attributes	87
Table 98 – ORIENTATION attributes	87
Table 99 – IMAGE attributes	88
Table 100 – PATH attribute	88
Table 101 – LINK attribute	88
Table 102 – Importing Device Description	90
Table 103 – Redefinition attributes	91
Table 104 – Redefinition rules for AXIS attributes	91

Table 105 – Redefinition rules for BLOB attributes.....	92
Table 106 – Redefinition rules for BLOCK_A attributes	93
Table 107 – Redefinition rules for BLOCK_B attributes	94
Table 108 – Redefinition rules for CHART attributes	94
Table 109 – Redefinition rules for COLLECTION attributes	95
Table 110 – Redefinition rules for COMMAND attributes	95
Table 111 – Redefinition rules for COMPONENT attributes	96
Table 112 – Redefinition rules for COMPONENT_FOLDER attributes	96
Table 113 – Redefinition rules for COMPONENT_REFERENCE attributes	97
Table 114 – Redefinition rules for COMPONENT_RELATION attributes	97
Table 115 – Redefinition rules for EDIT_DISPLAY attributes.....	98
Table 116 – Redefinition rules for FILE attributes	98
Table 117 – Redefinition rules for GRAPH attributes.....	99
Table 118 – Redefinition rules for GRID attributes	99
Table 119 – Redefinition rules for IMAGE attributes.....	100
Table 120 – Redefinition rules for INTERFACE attributes.....	100
Table 121 – Redefinition rules for LIST attributes.....	100
Table 122 – Redefinition rules for MENU attributes.....	101
Table 123 – Redefinition rules for METHOD attributes.....	101
Table 124 – Redefinition rules for PLUGIN attributes.....	102
Table 125 – Redefinition rules for RECORD attributes.....	102
Table 126 – Redefinition rules for REFERENCE_ARRAY attributes.....	103
Table 127 – Redefinition rules for RESPONSE_CODES attributes	103
Table 128 – Redefinition rules for SOURCE attributes	103
Table 129 – Redefinition rules for TEMPLATE attributes.....	104
Table 130 – Redefinition rules for VALUE_ARRAY attributes	104
Table 131 – Redefinition rules for VARIABLE attributes	105
Table 132 – Redefinition rules for VARIABLE_LIST attributes	106
Table 133 – Redefinition rules for WAVEFORM attributes.....	106
Table 134 – INTERFACE attributes	107
Table 135 – DECLARATION attributes	107
Table 136 – LIKE attributes.....	108
Table 137 – LIST attributes.....	108
Table 138 – TYPE attribute	109
Table 139 – CAPACITY attribute	109
Table 140 – COUNT attribute	110
Table 141 – MENU attributes	110
Table 142 – ITEMS attributes.....	111
Table 143 – ACCESS attribute	112
Table 144 – EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS attributes	113
Table 145 – STYLE attribute	115
Table 146 – METHOD attributes.....	118

Table 147 – Parameter types	119
Table 148 – ACCESS attributes	119
Table 149 – CLASS attributes	120
Table 150 – TYPE attributes	121
Table 151 – RECORD attributes	121
Table 152 – REFERENCE_ARRAY attributes	122
Table 153 – ELEMENTS attributes	122
Table 154 – REFRESH attributes	123
Table 155 – UNIT attributes	124
Table 156 – WRITE_AS_ONE attribute	124
Table 157 – RESPONSE_CODES attributes	125
Table 158 – SOURCE attributes	125
Table 159 – Y_AXIS attribute	127
Table 160 – TEMPLATE attributes	127
Table 161 – DEFAULT_VALUES attributes	128
Table 162 – VALUE_ARRAY attributes	128
Table 163 – NUMBER_OF_ELEMENTS attributes	129
Table 164 – TYPE attribute	129
Table 165 – VARIABLE attributes	130
Table 166 – CLASS attributes	131
Table 167 – TYPE attributes	132
Table 168 – DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER attributes	133
Table 169 – DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE attributes	136
Table 170 – BIT, ENUMERATED attributes	138
Table 171 – status_class attributes	139
Table 172 – ALL, AO, DV, TV attributes	140
Table 173 – Enumerated types attributes	140
Table 174 – Index type attributes	141
Table 175 – String types attributes	142
Table 176 – CONSTANT_UNIT attribute	143
Table 177 – DEFAULT_VALUE attribute	144
Table 178 – INITIAL_VALUE attribute	144
Table 179 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS attributes	145
Table 180 – POST_USERCHANGE_ACTIONS, POST_RQSTUPDATE_ACTIONS attributes	147
Table 181 – VARIABLE_LIST attributes	148
Table 182 – WAVEFORM attributes	149
Table 183 – TYPE attributes	149
Table 184 – XY attributes	150
Table 185 – YT attribute	151
Table 186 – HORIZONTAL attribute	151
Table 187 – VERTICAL attribute	152

Table 188 – EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS attributes	153
Table 189 – KEY_POINTS attributes	153
Table 190 – X_VALUES, Y_VALUES attributes.....	154
Table 191 – Y_AXIS attribute	155
Table 192 – CLASSIFICATION attributes	155
Table 193 – COMPONENT_PARENT attribute.....	157
Table 194 – COMPONENT_PATH attribute	157
Table 195 – DEFINITION attribute.....	158
Table 196 – EMPHASIS attributes.....	158
Table 197 – HANDLING attributes.....	158
Table 198 – HEIGHT/WIDTH attribute	159
Table 199 – HELP attribute	160
Table 200 – LABEL attribute	160
Table 201 – LINE_COLOR attributes.....	160
Table 202 – LINE_TYPE attribute.....	161
Table 203 – MEMBERS attributes	162
Table 204 – PROTOCOL attributes	162
Table 205 – RESPONSE_CODES attribute.....	163
Table 206 – SUPPLIED_INTERFACE attribute	163
Table 207 – VALIDITY attributes	164
Table 208 – PRIVATE attributes.....	164
Table 209 – VISIBILITY attributes	165
Table 210 – WRITE_MODE attributes	165
Table 211 – IDENTITY attribute	166
Table 212 – IF, SELECT conditional.....	167
Table 213 – Referencing an EDD instance	167
Table 214 – Referencing elements of VARIABLE	168
Table 215 – Referencing elements of RECORD.....	168
Table 216 – Referencing elements of VALUE_ARRAY	169
Table 217 – Referencing members of COLLECTION.....	169
Table 218 – Referencing members of REFERENCE_ARRAY	169
Table 219 – Referencing members of VARIABLE_LISTS	170
Table 220 – Referencing members of a BLOCK_A PARAMETERS	170
Table 221 – Referencing members of BLOCK_A PARAMETER_LISTS.....	170
Table 222 – Referencing members of BLOCK_A LOCAL_PARAMETER	171
Table 223 – Referencing BLOCK_A CHARACTERISTICS	171
Table 224 – Referencing members of FILE.....	171
Table 225 – Referencing elements of LIST	172
Table 226 – Referencing members of CHART.....	172
Table 227 – Referencing members of GRAPH	172
Table 228 – Referencing members of SOURCE.....	173
Table 229 – Referencing AXIS of a GRAPH, SOURCE, WAVEFORM	173
Table 230 – Referencing PARAMETERS of specific BLOCK_A instance.....	173

Table 231 – Referencing LOCAL_PARAMETERS of specific BLOCK_A instance.....	174
Table 232 – Referencing CHARACTERISTICS of specific BLOCK_A instance.....	174
Table 233 – Referencing CHARTS of specific BLOCK_A instance.....	175
Table 234 – Referencing LISTS of specific BLOCK_A instance	175
Table 235 – Referencing GRAPHS of specific BLOCK_A instance	176
Table 236 – Referencing GRIDS of specific BLOCK_A instance	176
Table 237 – Referencing MENUS of specific BLOCK_A instance	177
Table 238 – Referencing METHODS of specific BLOCK_A instance.....	177
Table 239 – Referencing a COMPONENT instance.....	178
Table 240 – Referencing a COMPONENT type.....	178
Table 241 – Referencing FILES of specific BLOCK_A instance	178
Table 242 – Referencing PLUGINS of specific BLOCK_A instance.....	179
Table 243 – String as a string literal	179
Table 244 – String as a string variable	179
Table 245 – String as an enumeration value.....	180
Table 246 – String as a dictionary reference.....	180
Table 247 – Referencing HELP and LABEL attributes of EDD instances.....	181
Table 248 – String operation	181
Table 249 – Format specifier.....	182
Table 250 – Primary expressions	183
Table 251 – Attribute values of VARIABLES.....	184
Table 252 – AXIS attribute values	185
Table 253 – BLOB attribute values	185
Table 254 – LIST attribute values.....	185
Table 255 – ARRAY attribute values.....	185
Table 256 – Unary expressions.....	185
Table 257 – Multiplicative operators.....	186
Table 258 – Additive operators.....	186
Table 259 – Shift operators.....	187
Table 260 – Relational operators.....	187
Table 261 – Equality operators.....	187
Table 262 – Text dictionary attributes.....	189
Table 263 – PLUGIN attributes.....	189
Table 264 – UUID attribute.....	190
Table 265 – BLOB attributes	190
Table A.1 – Conventions for integer constants.....	195
Table A.2 – Using escape sequences in string literals.....	196
Table A.3 – Language code examples for string literals	197
Table A.4 – Precedence and associativity for EDDL operators	198
Table A.5 – Operations for VARIABLES or METHOD local variables.....	200
Table A.6 – EDDL keywords.....	201
Table D.1 – Profile selection tables	304
Table D.2 – EDDL Formal Definition profile tables	304

Table D.3 – Contents of selection tables	304
Table D.4 – EDDL element selection for PROFIBUS and PROFINET	305
Table D.5 – EDDL element selection for FOUNDATION fieldbus	312
Table D.6 – EDDL element selection for HCF	319
Table D.7 – EDDL element selection for Communication Servers	327
Table D.8 – METHOD DEFINITION data types	334
Table D.9 – VARIABLE TYPES	335
Table D.10 – DATE coding	336
Table D.11 – DATE_AND_TIME coding	337
Table D.12 – DURATION coding	337
Table D.13 – TIME coding	338
Table D.14 – TIME_VALUE coding (four octets)	338
Table D.15 – TIME_VALUE coding (eight octets)	338
Table D.16 – PACKED_ASCII coding	339
Table D.17 – BOOLEAN coding	339

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**FUNCTION BLOCKS (FB) FOR PROCESS CONTROL AND
ELECTRONIC DEVICE DESCRIPTION LANGUAGE (EDDL) –****Part 3: EDDL syntax and semantics**

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 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 61804-3 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

This third edition cancels and replaces the second edition published in 2010. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

- Builtins and their profiles removed and relocated into IEC 61804-5.
- The following extensions are integrated in the EDDL specification to meet FDI requirements:
 - New constructs BLOB, PLUGIN.
 - Component construct for communication server requirements (additional attributes).

- Extension of the class attribute.
- New attributes PRIVATE, VISIBILITY, WRITE_MODE
- The following changes will be integrated in the EDDL based on EDDL harmonization:
 - Removed some unused features.
 - Harmonized some profile features.

The text of this standard is based on the following documents:

FDIS	Report on voting
65E/451/FDIS	65E/462/RVD

Full information on the voting for the approval of this 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.

Headings ending with '(void)' are used to retain the numbering of previous editions.

A list of all parts in IEC 61804 series, published under the general title *Function blocks (FB) for process control and Electronic Device Description Language (EDDL)*, can be found on the IEC website.

Future standards in this series will carry the new general title as cited above. Titles of existing standards in this series will be updated at the time of the next edition.

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 EDDL fills the gap between the conceptual function block specification of IEC 61804-2 and a product implementation. It allows the manufacturers to use the same description method for devices based on different technologies and platforms. Figure 1 shows these aspects.

IEC 61804 has the general title "Function blocks (FB) for process control and Electronic Device Description Language (EDDL)" and consists of the following parts:

Part 2: Specification of FB concept

Part 3: EDDL syntax and semantics

Part 4: EDD interpretation

Part 5: EDDL Built-in library

Part 6: Meeting the requirements for integrating fieldbus devices in engineering tools for field devices

This part of IEC 61804 has integrated some parts of IEC TS 61804-1:2003, which was withdrawn in January 2013.

The EDDL may also be used for the description of product properties in other domains such as industrial automation. Industrial automation may include devices such as generic digital and analog input/output modules, motion controllers, human-machine interfaces, sensors, closed-loop controllers, encoders, hydraulic valves, and programmable controllers.

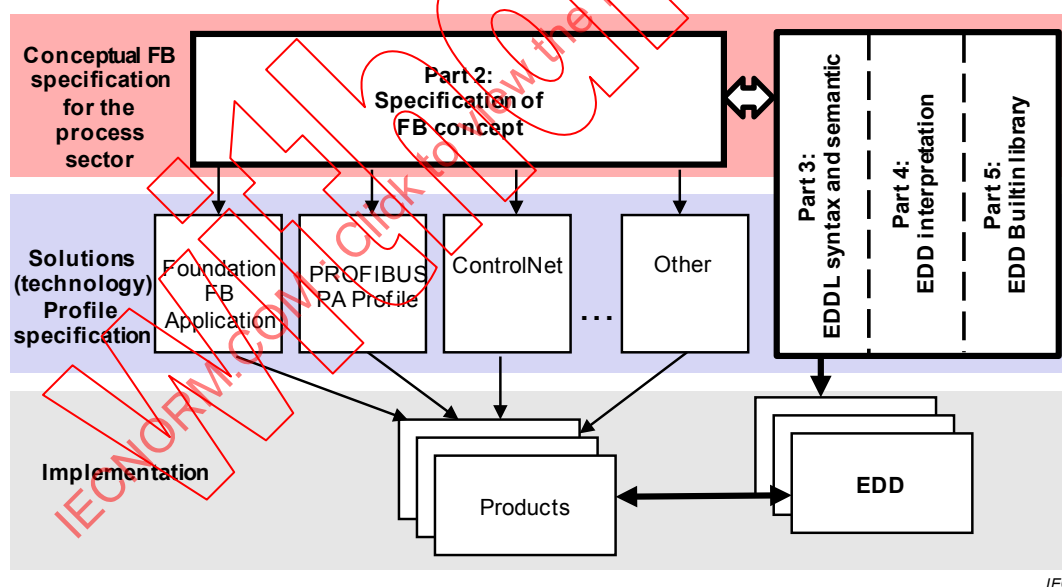


Figure 1 – Position of IEC 61804 in relation to other standards and products

FUNCTION BLOCKS (FB) FOR PROCESS CONTROL AND ELECTRONIC DEVICE DESCRIPTION LANGUAGE (EDDL) –

Part 3: EDDL syntax and semantics

1 Scope

This part of IEC 61804 specifies the Electronic Device Description Language (EDDL) technology, which enables the integration of real product details using the tools of the engineering life cycle.

This part of IEC 61804 specifies EDDL as a generic language for describing the properties of automation system components. EDDL is capable of describing

- device parameters and their dependencies;
- device functions, for example, simulation mode, calibration;
- graphical representations, for example, menus;
- interactions with control devices;
- graphical representations:
 - enhanced user interface,
 - graphing system;
- persistent data store.

EDDL is used to create Electronic Device Description (EDD) for example concrete devices, common usable profiles or libraries. This EDD is used with appropriate tools to generate an interpretative code to support parameter handling, operation, and monitoring of automation system components such as remote I/Os, controllers, sensors, and programmable controllers. Tool implementation is outside the scope of this standard.

This part of IEC 61804 specifies the semantic and lexical structure in a syntax-independent manner. A specific syntax is defined in Annex A, but it is possible to use the semantic model also with different syntaxes.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60050 (all parts), *International Electrotechnical Vocabulary* (available at <http://www.electropedia.org>)

IEC 61804-2:–1, *Function blocks (FB) for process control and Electronic Device Description Language (EDDL) – Part 2: Specification of FB concept and Electronic Device Description Language (EDDL)*

¹ To be published.

IEC 61804-5, *Function blocks (FB) for process control and Electronic Device Description Language (EDDL) – Part 5: EDDL Builtin library*

IEC 62541-4, *OPC unified architecture – Part 4: Services*

ISO/IEC 2375, *Information technology – Procedure for registration of escape sequences and coded character sets*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 9834-8, *Information technology – Procedures for the operation of object identifier registration authorities – Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers*

ISO/IEC 9899, *Information technology – Programming languages – C*

ISO/IEC 10646-1, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

ISO 639 (all parts), *Codes for the representation of names of languages*

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*

RFC 3629, *UTF-8, User Datagram Protocol*, available at <http://www.ietf.org/rfc/rfc0768.txt>

W3C Cascading Style Sheets Level 2 Specification <http://www.w3.org/TR/CSS2>

3 Terms, definitions, abbreviated terms and acronyms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 60050-351, in IEC 61804-2, as well as the following apply.

3.1.1

Builtin

predefined subroutine executed by the EDD application, e.g. for communication and display purposes

3.1.2

device

independent physical entity capable of performing one or more specified functions in a particular context and delimited by its interfaces

[SOURCE: IEC 61499-1:2012, 3.29]

3.1.3**EDD application**

program using the EDD, or any translated form, which offers functionality such as communication representation, data representation, graphical representation, etc.

3.1.4**EDDL processor**

processor or program, which translates the EDD into an executable form that can be processed by an EDD application

3.1.5**EDDL profile**

selection of the supported elements of the EDDL lexical structure including the syntax definitions for a number of specific consortia

3.1.6**Electronic Device Description Language****EDDL**

formal language for describing devices and communication components

3.1.7**Electronic Device Description****EDD**

data collection containing the device parameter(s), its(their) dependencies, its(their) graphical representation and a description of the data sets which are transferred

Note 1 to entry: The Electronic Device Description is created using the EDDL.

3.1.8**Electronic Device Description compiler**

compiler which translates the EDD source in an internal format that is used by the EDD Interpreter

3.1.9**Electronic Device Description Interpreter****EDDI**

Interpreter which uses the EDD source or an internal format that is given by the EDDL compiler to provide the EDD information to the EDD user

3.1.10**function block****function block instance**

software functional unit comprising an individual, named copy of a data structure and associated operations specified by a corresponding FB type

Note 1 to entry: Typical operations of an FB include modification of the values of the data in its associated data structure.

[SOURCE: IEC 61804-2;–, 3.1.32]

3.1.11**hardware**

physical equipment, as opposed to programs, procedures, rules and associated documentation

3.1.12**preprocessor**

part of the run-time environment that transforms the information given in an EDD

3.1.13**preprocessor directives**

description of conditions for filtering the EDD code before compilation or interpretation

Note 1 to entry: For example, a preprocessor directive providing the facility to define names for constants or to write macros to make code easier to read.

3.1.14**software**

intellectual creation comprising the programs, procedures, rules and any associated documentation pertaining to the operation of a system

[SOURCE: IEC 61499-1:2012, 3.92]

3.2 Abbreviated terms and acronyms

ASCII	American Standard Code for Information Interchange according to ISO/IEC 10646-1
CP	Communication Profile
CPF	Communication Profile Family
EDD	Electronic Device Description
EDDL	Electronic Device Description Language
EUC	Extended Unicode according to ISO/IEC 2022
FB	Function block
FF	Fieldbus Foundation ²
HCF	HART® ³ Communication Foundation
HMI	Human-machine interface
I/O	Input/Output
ID	Identifier
OSI	Open Systems Interconnection
PI	PROFIBUS and PROFINET ⁴ International
PDU	Protocol data unit
PNO	PROFIBUS Nutzerorganisation

4 Conformance statement

A statement of conformance to a profile of this part of IEC 61804 shall be stated⁵ as

– compliance to IEC 61804-3:–, Annex D.n <Type>

² FOUNDATION™ Fieldbus is the trade name of the consortium Fieldbus Foundation (non-profit organization). This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the trade name. Use of the trade names requires permission from the trade name holder.

³ HART® is a registered trademark of the consortium HART Communication Foundation (HCF) (non-profit organization). This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the trade name. Use of the trade names requires permission from the trade name holder.

⁴ PROFIBUS and PROFINET are trade names of PROFIBUS Nutzerorganisation e.V. (PNO) as part of PROFIBUS and PROFINET International. PNO is a non-profit trade organization to support the fieldbus technologies PROFIBUS and PROFINET. This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the tradename. Use of the trade names requires permission from the trade name holder.

⁵ In accordance with the ISO/IEC Directives.

where the Type within the angle brackets < > is optional and the angle brackets are not to be included.

The Type may be, for example, the name of the communication protocol used such as PROFIBUS, Foundation Fieldbus, or HART Communication Foundation, or in an encoded way according to the Communication Profiles specified in IEC 61784-1 or in IEC 61784-2.

Product standards shall not include any conformity assessment aspects (including QM provisions), either normative or informative, other than provisions for product testing (evaluation and examination).

5 Conventions

5.1 General

In IEC 61804 series all keywords are written in uppercase letters. An EDDL element is written in lowercase letters to address the semantic of the whole element.

The EDDL is generally described using lexical structures in which the elements and the presence of fields are specified.

5.2 Conventions for lexical structure

5.2.1 ABC field1, field2

ABC is a lexical element. This element shall be coded in a concrete syntax. It is not required to code this element with the name “ABC”. It is also possible to code this element, for example, with a tag number.

Field1 and field2 are fields of the lexical element ABC. Each field is mandatory and may have more than one attribute. If a field has attributes, the presence of the attributes is specified in a table. A comma separates field1 and field2. The comma is a lexical element and is not coded explicitly.

If a field has additional attributes, the attributes are defined in a table. The table layout and the possible usage qualifiers are shown in Table 1.

Table 1 – Field attribute descriptions

Usage	Attribute	Description
m	yyy	The presence of this attribute is mandatory.
o	xxx	The presence of this attribute is optional.
s	z1	The presence of this attribute is selectable with other attributes, which are also marked with “s” in the “usage” column. Only one of the selectable attributes (z1 or z2) is present.
s	z2	The presence of this attribute is selectable with other attributes, which are also marked with “s” in the “usage” column. Only one of the selectable attributes (z1 or z2) is present.
c	uuu	The presence of this attribute is conditional and it is only present if the condition is true.

The characters in the “usage” column have the following meanings:

m: this attribute is mandatory and shall be present.

o: this attribute is optional and need not be present.

s: this attribute is a selection. One, and only one, of the fields marked with “s” (z1 or z2) shall be present.

c: this attribute is conditional. The condition is described in the “description” column.

In the “attribute” column, if more than one attribute exists, which have the same usage, the attributes are sorted in alphabetical order.

5.2.2 ABC field1+

The plus (+) behind field1 is used to indicate that field1 is used at least once. It may be used more than once.

5.2.3 ABC field2*

The asterisk (*) behind field2 is used to indicate that field2 is optional and need not be used. If it is used it may be used more than once.

5.2.4 ABC [field1, field2]+

The elements field1 and field2 in the square brackets [] are an unsorted list. The plus (+) behind the closing bracket indicates that field1 and field2 are used at least once and may be used more than once as group.

5.2.5 ABC field1, (field2, field3)<exp>

<exp> indicates that field2 and field3 are used in conjunction with the conditional expression, (conditional constructs are specified in 7.37). The expression <exp> refers only to the fields within the preceding curved brackets (). Usage of conditional expressions is optional.

6 EDD and EDDL model

6.1 Overview of EDD and EDDL

An Electronic Device Description (EDD) contains all device parameters of an automation system component. The technology used to describe an EDD is called Electronic Device Description Language (EDDL). The EDDL provides a set of scalable basic language elements to handle simple, complex or modular devices. The EDDL is a descriptive language based on an ASCII format with clear separation between data and program.

Data in a text field, which is marked with a country code like Japanese, may use a multi-byte code.

6.2 EDD architecture

From the viewpoint of the ISO/OSI model (ISO/IEC 7498-1), an EDD is above Layer 7. However, the EDD application uses the communication system to transfer its information. An EDD contains constructs that support mapping to a supporting communication system.

The device manufacturer defines the objects, which are reflected by the logical representation of the objects within an EDD application. For that reason, EDDL has language elements, which map the EDD data to the data representation of the communication system, so that the user of an EDD does not need to know the physical location (address) of a device object.

EDD describes the management of information to be displayed to the user. The specific representation of such visualization is not part of EDD or EDDL definitions.

6.3 Concepts of EDD

The manufacturer of a device or of an automation system component describes the properties of the device by using the EDDL. The resulting EDD contains information such as:

- description of the device parameters;

- description of parameter dependencies;
- logical grouping of the device parameters;
- selection and execution of supported device functions;
- description of the transferred data sets.

Depending on the required usage, the EDD may be physically located

- in a device;
- in an external data storage medium such as a compact disk, floppy or a server;
- partially distributed in the device and an external storage medium.

EDD supports text strings (common terms, phrases etc.) in more than one language (English, German, French, etc.). Text strings may be stored in separate dictionaries. There may be more than one dictionary for one EDD.

An EDD implementation includes sufficient information about the target device, for example, manufacturer, device type, revision, etc. This is used to match a specific EDD to a specific device.

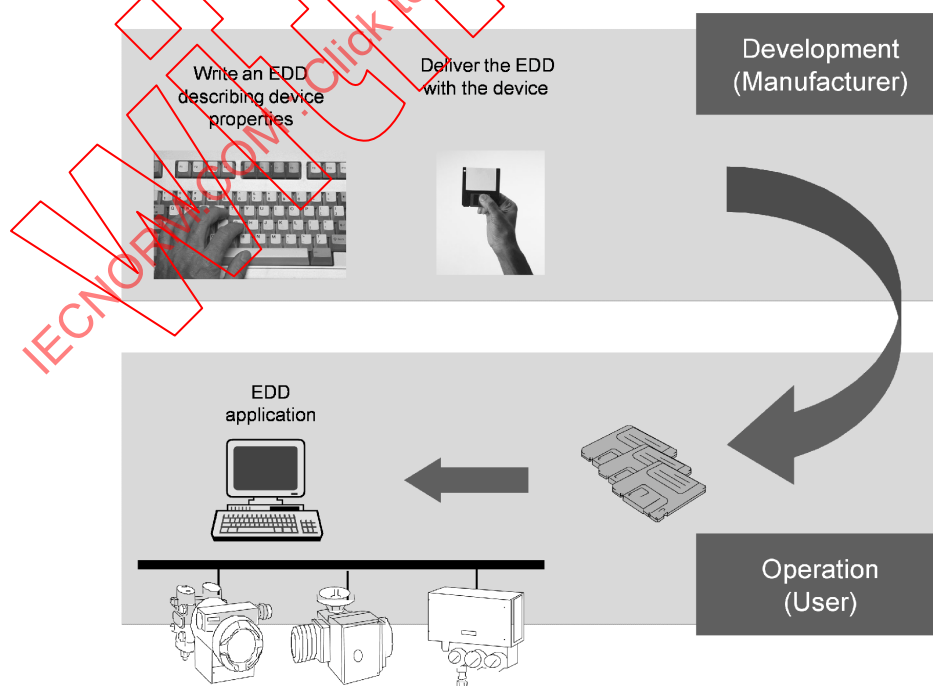
6.4 Principles of the EDD development process

6.4.1 General

The creation of an EDD is a three-stage process: EDD source generation, EDD preprocessing and EDD compilation.

6.4.2 EDD source generation

The EDD generation process is shown in Figure 2.



IEC

Figure 2 – EDD generation process

The device manufacturer writes an appropriate EDD for his device and delivers both (EDD and device). If the automation system supports the EDD method, a system integration can be made by the user.

The EDDs for devices may be embedded in the device memory or delivered using separate storage media or downloaded from an appropriate network server. The EDD is 'interpreted' or 'browsed' by an EDD application. EDDs are normally stored as source files or preprocessed files.

6.4.3 EDD preprocessing

In the preprocessing stage an EDD preprocessor generates a consistent EDD representation suitable for final compilation.

Preprocessing supports, for example, substitution of definitions and inclusion of external text. The output of the preprocessor is a complete EDD without any preprocessor directive.

6.4.4 EDD compilation

The EDD compilation stage produces an EDD application internal representation from a preprocessed EDD to be used in the EDD application.

6.5 Interrelations between the lexical structure and formal definitions

The lexical structure of EDDL and its elements are described in Clause 7. Formal definitions and syntax for each EDDL element are given in Annex A. The lexical structure and its formal description use the same name.

NOTE Instead of the specified formal definitions and syntax in Annex A, another specification may be developed as an additional option.

6.6 Builtins

Builtins are predefined subroutines which are executed in the EDD application.

EXAMPLE A hand-held terminal is a simple device having a small display and limited cursor functions. For this type of device, a Builtin could be specified to provide display entry using only up/down right/left cursor actions.

The library of Builtins is defined in IEC 61804-5.

6.7 Profiles

EDDL is a harmonized specification of existing legacy EDD concepts. Concrete EDD applications use a subset of the EDDL specification. Selection of EDDL elements and Builtins is made from the profiles defined in Annex D.

In addition to EDD profiles, implementing consortia also publish "Device Profiles", which are used to support the interchangeability of compliant devices. These Device Profiles may be described using the EDDL specified in this document.

7 Electronic Device Description Language (EDDL)

7.1 Overview

7.1.1 EDDL features

The EDDL is a structured and interpretative language for describing device properties. Also the interactions between the devices and the EDD run-time environment are incorporated in the EDDL. The EDDL provides a set of language elements for these purposes.

For a specific EDD implementation it is not necessary to use all of the elements provided by the language.

Compatible subsets of EDDL are permitted and may be specified using profiles (for example, choice of constructs, number of recursions, and selection of options). Profiles supported by some industrial consortia are specified in Annex D. EDD developers are required to identify within each device details as to which profile has been used.

7.1.2 Syntax representation

The specification of the EDDL language in Clause 7 of this part of IEC 61804 uses an abstract syntax. The abstract lexical structure in 7.1 is converted into specific syntaxes specified in Annex A by the element name (for example, VARIABLE in the lexical structure equates to the keyword “VARIABLE” in Annex A).

Beside the defined syntax in this part of IEC 61804, it is possible to use other syntax definitions, which may be added in the future to allow other features and representations for future progress.

7.1.3 EDD language elements

The language is structured with the following language elements.

- identification element;
- basic construction elements;
- special elements.

Frequently used text strings, for example, help text and multi-lingual lists of labels should be separated in a text dictionary (see 7.41).

Identification information shall be the first entry in every EDD file and shall appear only once. The identification information uniquely identifies the version of EDDL used, together with the specific device type, model codes and revision details covered by the EDD file.

7.1.4 Basic construction elements

7.1.4.1 General

These basic constructs have been specified to support descriptions of devices used within industrial control applications, together with their properties and functionality.

Some constructs have similar names and functions while differing in their detailed specification. This additional variety has been included to ensure compatibility with several existing description languages. Appropriate mapping cross-references are given by profiles.

Each of the basic constructs has a set of attributes associated with it. Attributes can also have subattributes, which refine the definition of the attribute and hence the definition of the construct itself.

The definition of an attribute may be static or dynamic. A static attribute definition never changes, while a dynamic attribute definition may change in order to accommodate parameter value changes.

7.1.4.2 AXIS

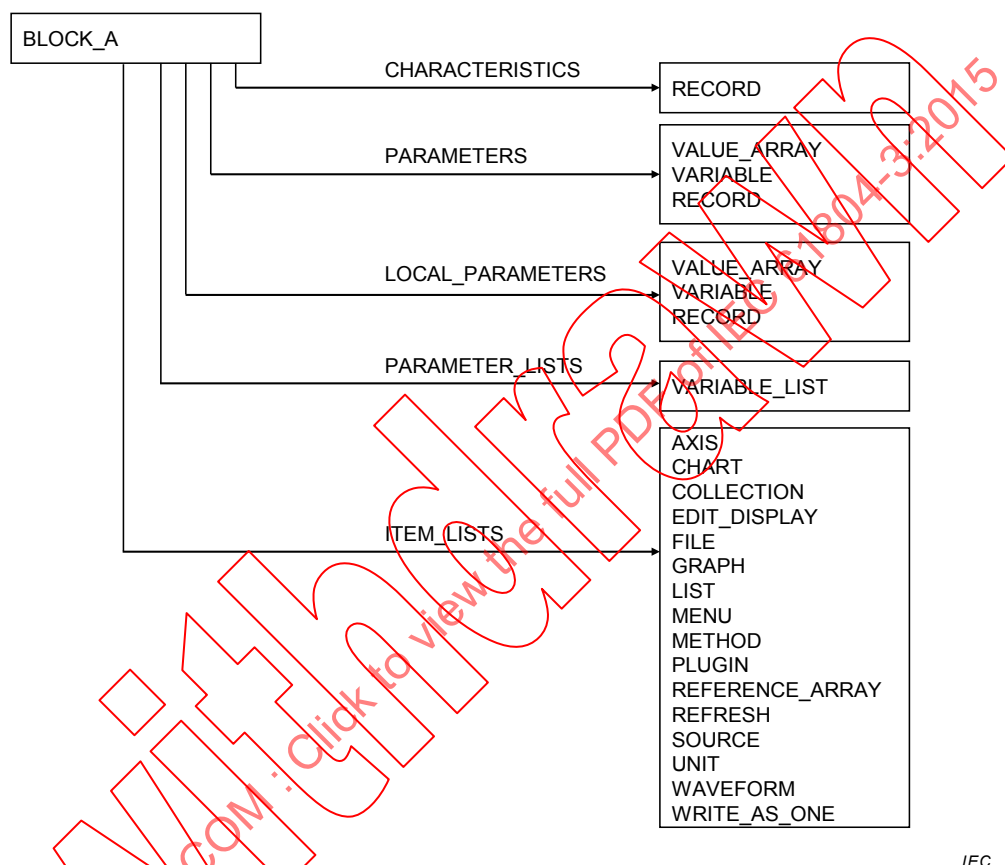
AXIS describes the axis of a CHART or GRAPH (see 7.3).

7.1.4.3 BLOB

BLOB describes a Binary Large Object used for transferring binary data to or from a device (see 7.43).

7.1.4.4 BLOCK_A

BLOCK_A is a logical grouping of CHARACTERISTICS, PARAMETERS, PARAMETER_LISTS, and ITEM_LISTS, see Figure 3. To access one item of BLOCK_A, the instance of the block should be used (see 7.4.1 and Figure 3).



IEC

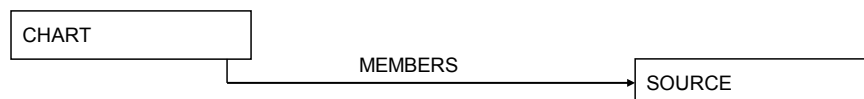
Figure 3 – BLOCK_A

7.1.4.5 BLOCK_B

BLOCK_B is used to structure variables in instances of logical block types. For accessing a VARIABLE within a BLOCK_B construct, the basic construct COMMAND is used to provide the relative addressing (see 7.4.2).

7.1.4.6 CHART

CHART describes a chart used to display data from a device in the EDD application (see 7.5 and Figure 4).



IEC

Figure 4 – CHART

7.1.4.7 COLLECTION

COLLECTION describes logical groupings of data (see 7.6 and Figure 5).

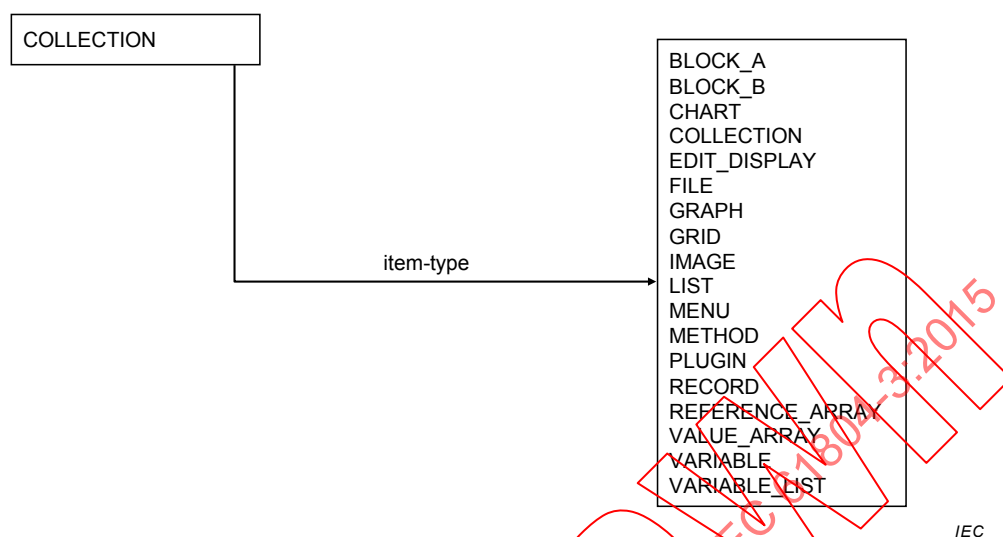


Figure 5 – COLLECTION

7.1.4.8 COMMAND

COMMAND describes the structure and the addressing of the variables in the device (see 7.7 and Figure 6).

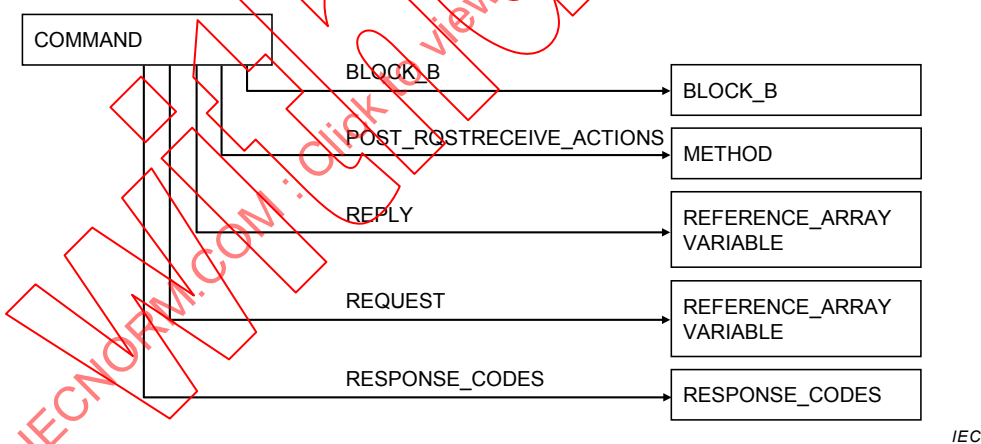


Figure 6 – COMMAND

7.1.4.9 COMPONENT

COMPONENT describes the configuration behavior of the device that is described in the EDD (see 7.8 and Figure 7).

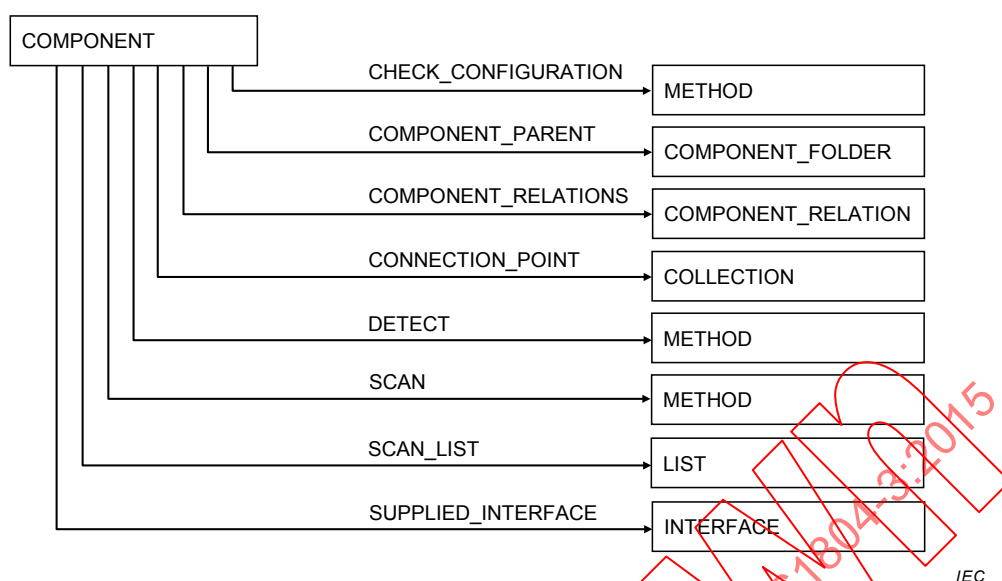


Figure 7 – COMPONENT

7.1.4.10 COMPONENT_FOLDER

COMPONENT_FOLDER describes a folder in a catalog. Folders allow groupings of components in a catalog (see 7.9 and Figure 8).



Figure 8 – COMPONENT_FOLDER

7.1.4.11 COMPONENT_REFERENCE

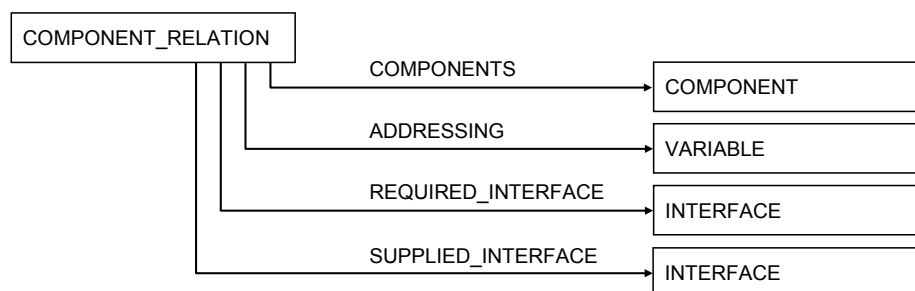
COMPONENT_REFERENCE refers to a **COMPONENT** or a **COMPONENT_FOLDER** (see 7.10 and Figure 9).



Figure 9 – COMPONENT_REFERENCE

7.1.4.12 COMPONENT_RELATION

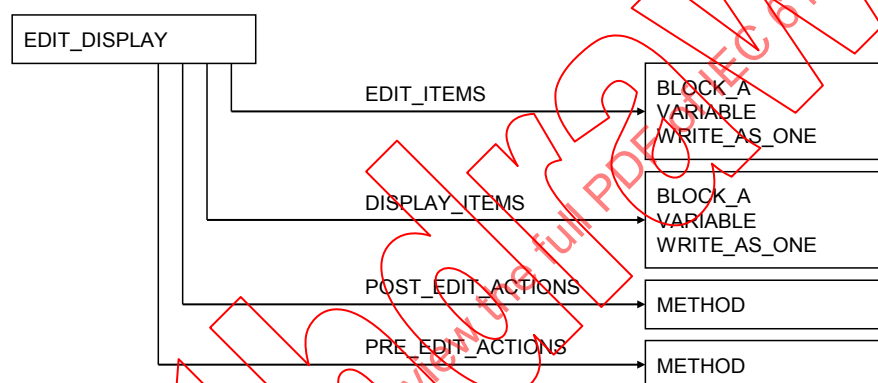
COMPONENT_RELATION specifies a relation between **COMPONENTS**, for example a module and one or more other modules (see 7.11 and Figure 10).



IEC

Figure 10 – COMPONENT_RELATION**7.1.4.13 EDIT_DISPLAY**

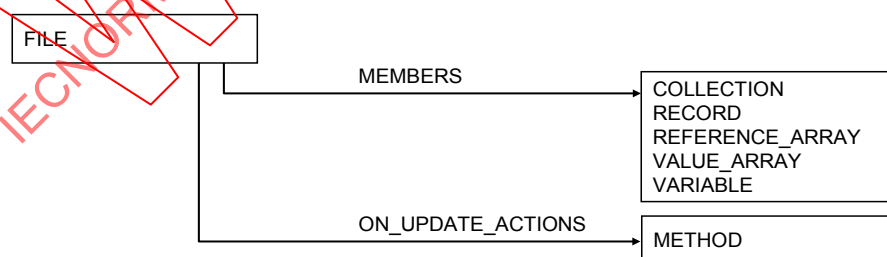
EDIT_DISPLAY describes how the data will be represented to a user by a display device (see 7.14 and Figure 11).



IEC

Figure 11 – EDIT_DISPLAY**7.1.4.14 FILE**

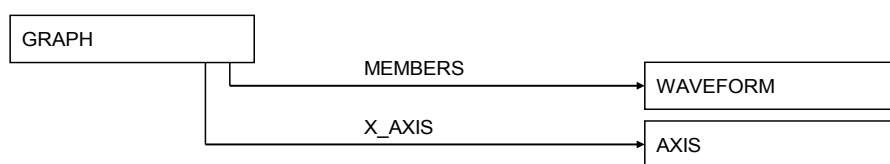
FILE describes an area of persistent storage (see 7.15 and Figure 12).



IEC

Figure 12 – FILE**7.1.4.15 GRAPH**

GRAPH describes a graph used to display data from a device in the EDD application (see 7.16 and Figure 13).



IEC

Figure 13 – GRAPH

7.1.4.16 GRID

GRID describes a matrix used to display data from a device in the EDD application (see 7.17 and Figure 14).

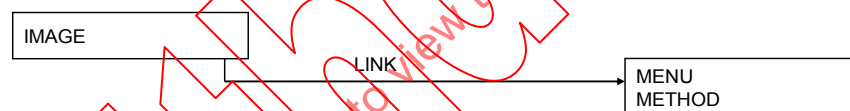


IEC

Figure 14 – GRID

7.1.4.17 IMAGE

IMAGE describes a visual image (see 7.18 and Figure 15).



IEC

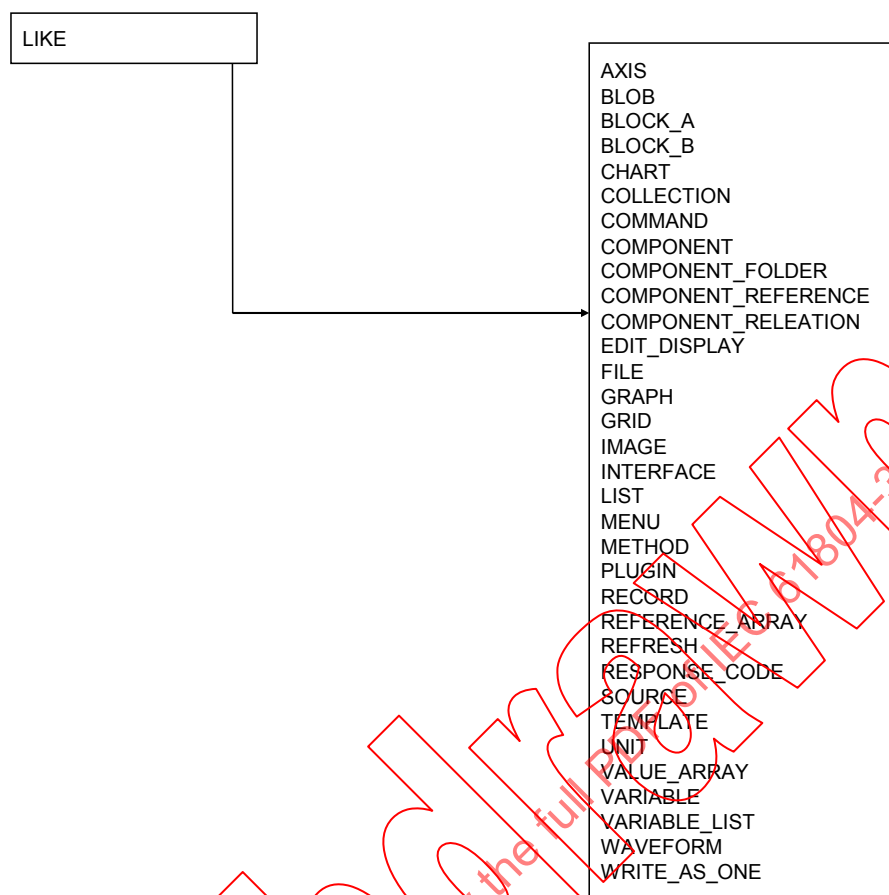
Figure 15 – IMAGE

7.1.4.18 IMPORT

IMPORT is used to import an EDD and to modify its existing definitions (see 7.19).

7.1.4.19 LIKE

LIKE creates a new instance of an existing instance of a basic construct. The attributes of the new instance may be redefined, added or deleted (see 7.21 and Figure 16).

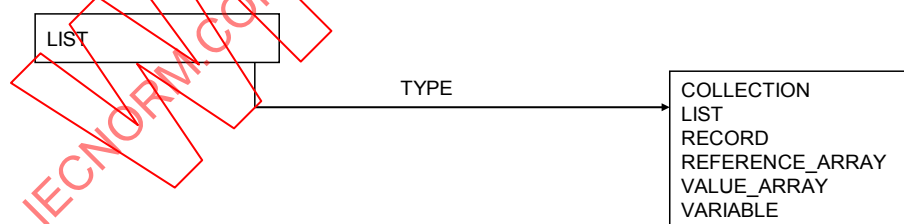


IEC

Figure 16 – LIKE

7.1.4.20 LIST

LIST describes a sequence of elements (see 7.22 and Figure 17).



IEC

Figure 17 – LIST

7.1.4.21 MENU

MENU describes how the data will be presented to a user by an EDD application (see 7.23 and Figure 18).

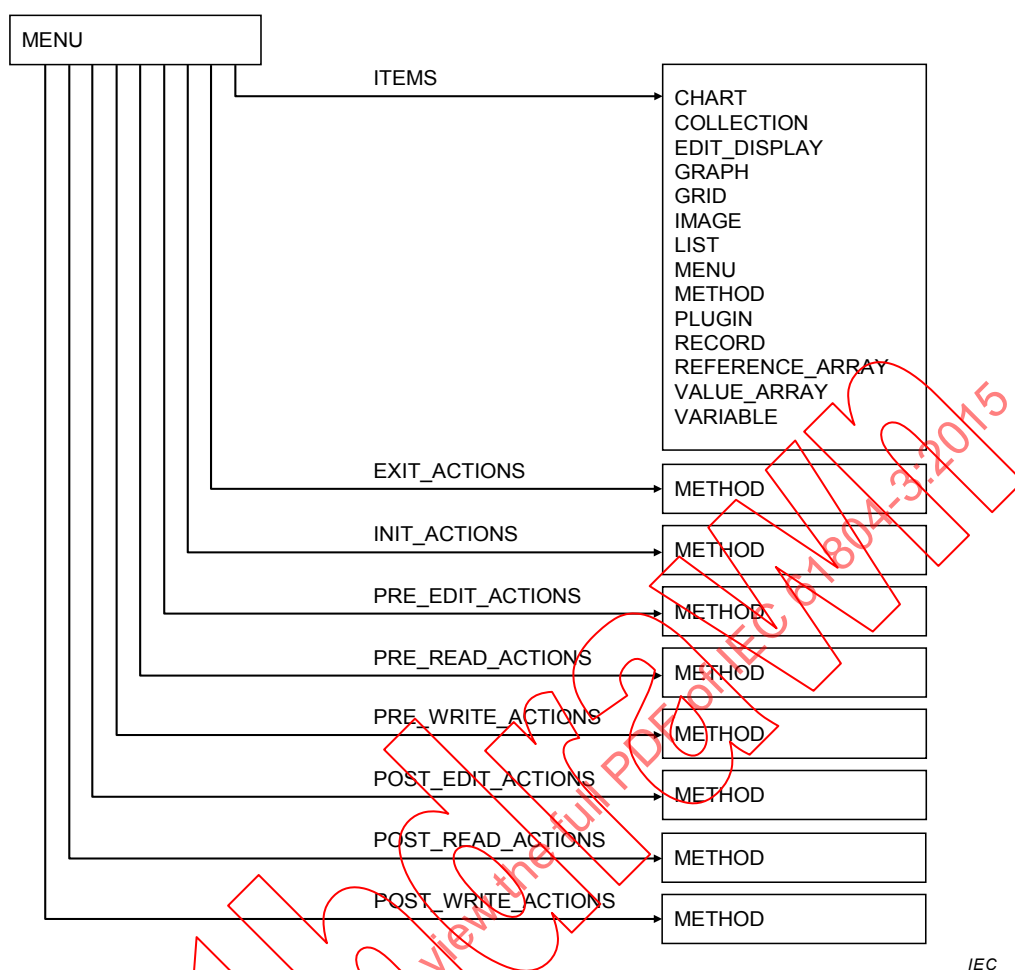


Figure 18 – MENU

7.1.4.22 METHOD

METHOD describes the execution of a complex sequence of event interactions that shall take place between system devices such as display units, configurators and devices (see 7.24).

7.1.4.23 PLUGIN

PLUGIN describes a User Interface Plug-in (UIP) as specified in 7.42.

7.1.4.24 RECORD

RECORD describes the data contained in the device (see 7.26 and Figure 19).

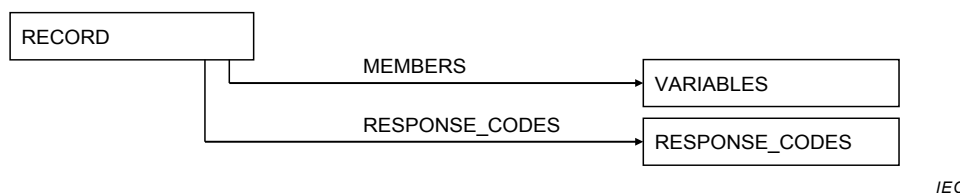


Figure 19 – RECORD

7.1.4.25 REFERENCE_ARRAY

REFERENCE_ARRAY is a set of EDD items of the same EDDL item type (for example, VARIABLE or MENU). An item can be referenced in the EDD via the REFERENCE_ARRAY name and the index associated with the item (see 7.27 and Figure 20).

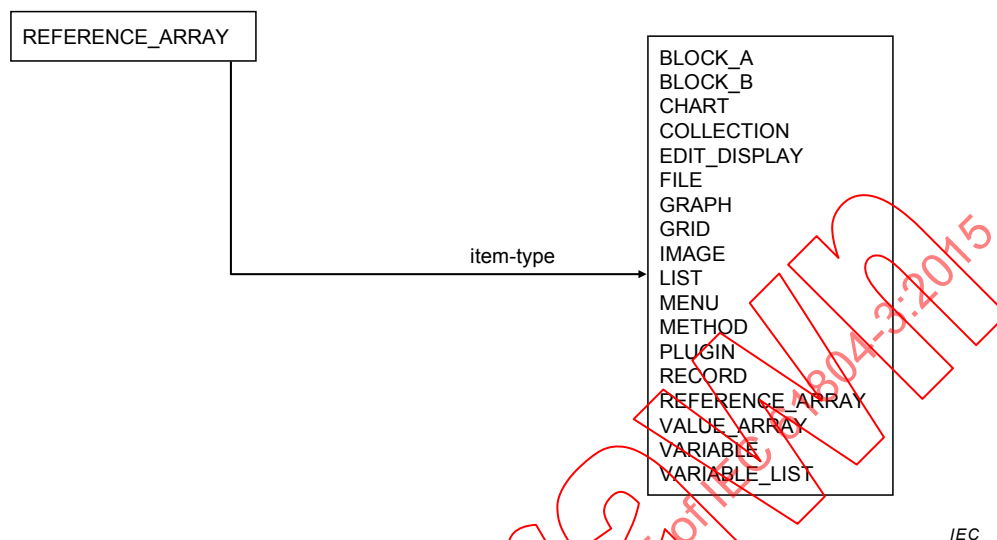


Figure 20 – REFERENCE_ARRAY

7.1.4.26 Relations

7.1.4.26.1 REFRESH

REFRESH describes relationships between LISTS, RECORDs, VALUE_ARRAYs and VARIABLEs (see 7.28.1 and Figure 21).

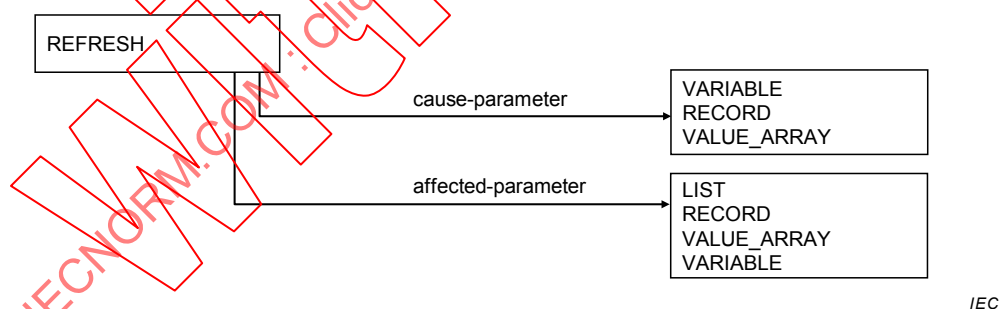


Figure 21 – REFRESH

7.1.4.26.2 UNIT

UNIT describes relationships between AXISs, VALUE_ARRAYs and VARIABLEs (see 7.28.2 and Figure 22).

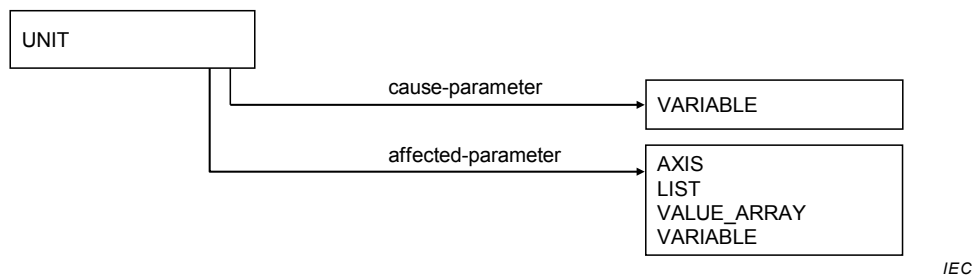


Figure 22 – UNIT

7.1.4.26.3 WRITE_AS_ONE

WRITE_AS_ONE describes relationships between VARIABLES, RECORDs, and VALUE_ARRAYs (see 7.28.3 and Figure 23).



Figure 23 – WRITE_AS_ONE

7.1.4.27 RESPONSE_CODES

RESPONSE_CODES specifies the list of response code elements that shall be used by the EDD application to inform the user about the communication result, for example error, warning (see 7.29).

7.1.4.28 SOURCE

SOURCE describes a source of data values for a CHART (see 7.30 and Figure 24).

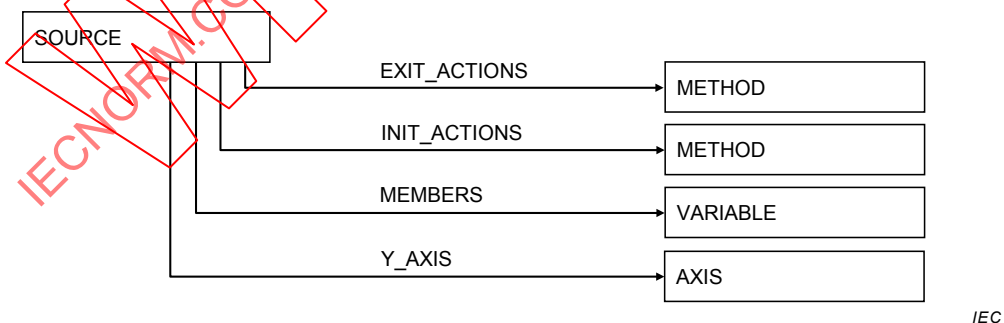
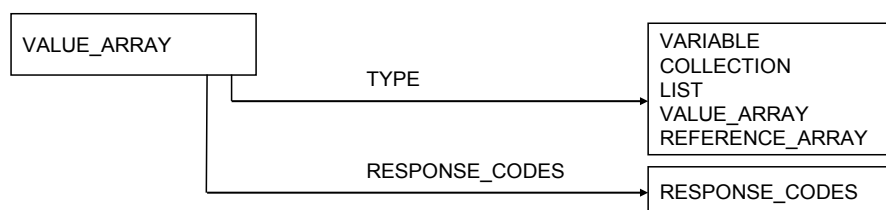


Figure 24 – SOURCE

7.1.4.29 VALUE_ARRAY

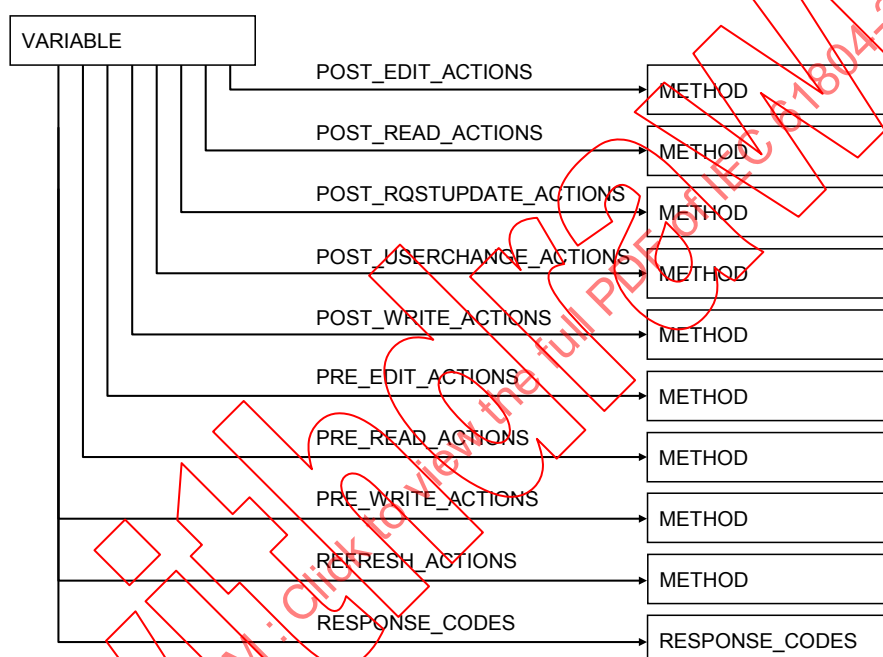
VALUE_ARRAY describes logical groups of EDDL variables (see 7.32 and Figure 25).



IEC

Figure 25 – VALUE_ARRAY**7.1.4.30 VARIABLE**

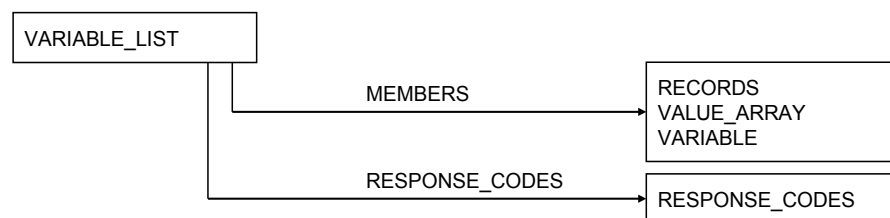
VARIABLE describes the data contained in the device (see 7.33 and Figure 26).



IEC

Figure 26 – VARIABLE**7.1.4.31 VARIABLE_LIST**

VARIABLE_LIST describes logical groupings of data contained in the device that may be communicated as a list (see 7.34 and Figure 27).



IEC

Figure 27 – VARIABLE_LIST

7.1.4.32 WAVEFORM

WAVEFORM describes a data set that may be displayed by a GRAPH (see 7.35 and Figure 28).

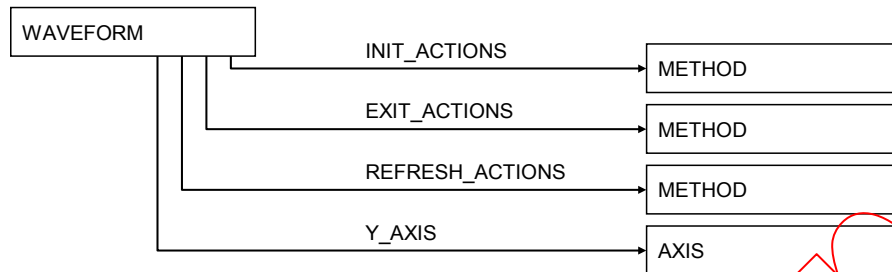


Figure 28 – WAVEFORM

7.1.5 Common attributes

The following common attributes are used in many basic constructs:

- CLASSIFICATION
- COMPONENT_PARENT
- COMPONENT_PATH
- DEFINITION
- EMPHASIS
- HANDLING
- HEIGHT
- HELP
- IDENTITY
- LABEL
- LINE_COLOR
- LINE_TYPE
- MEMBERS
- PRIVATE
- PROTOCOL
- RESPONSE_CODES
- SUPPLIED_INTERFACE
- VALIDITY
- VISIBILITY
- WIDTH
- WRITE_MODE

7.1.6 Special elements

The special elements are additional EDDL mechanisms to support file handling, multiple instances and modification of EDDL basic elements, such as:

- **Conditional Expressions** specify attributes values, which are dependent on the values of variables at run-time.

- **Reference** is used throughout a device description to refer to other items within the same EDDL.
- **Expression** is a logical or mathematical operation on variable values in order to modify attribute values of a basic construct at run-time.

7.1.7 Rules for instances

Each instance of a basic construct shall be identified by an identifier of type string. This identifier shall be unique within the complete EDD. All EDDL constructs represent actual instances of device information. If a device contains two or more pieces of information which have the same structure the appropriate construct shall be repeated in the EDD with different identifiers. Alternatively the LIKE construct may be used to create a copy of the construct with a different identifier. The identifier in the lexical definition part is not described within each construct again.

7.1.8 Rules for a list of VARIABLES

A list of VARIABLEs may contain VARIABLEs, elements of composed types or composed types. Composed types are:

- BLOCK_A PARAMETERS
- COLLECTIONS
- RECORDS
- REFERENCE_ARRAY
- VALUE_ARRAY
- VARIABLE_LISTS

Instead of referencing every single member of a composed type, a reference can be made to the type instance.

7.2 EDD identification information

7.2.1 General structure

Purpose

EDD identification information uniquely identifies the device description of a specific device type from a device manufacturer. It also specifies the EDDL version and EDDL profile used.

This identification information shall be the first entry in every EDD and shall appear only once. Strict order of the keywords of the EDD identification information is not necessary.

Lexical structure

DD_REVISION, DEVICE_REVISION, DEVICE_TYPE, EDD_PROFILE, EDD_VERSION,
MANUFACTURER, MANUFACTURER_EXT

7.2.2 Specific attributes

7.2.2.1 DD_REVISION

Purpose

The DD_REVISION attribute specifies a unique code for the EDD revision of this device, as defined by the manufacturer.

The manufacturer should change the revision number each time a new device description is issued for a device.

There can be multiple EDDs for a particular revision of a particular device, in which case the DD_REVISION distinguishes them.

Lexical structure

DD_REVISION integer

The attribute of DD_REVISION is specified in Table 2.

Table 2 – DD_REVISION attribute

Usage	Attribute	Description
m	integer	Two-octet unsigned integer containing the code for a specific EDD file revision.

7.2.2.2 DEVICE_REVISION

Purpose

The DEVICE_REVISION attribute specifies a unique code for the device revision of the device, as defined by the manufacturer.

Lexical structure

DEVICE_REVISION integer

The attribute of DEVICE_REVISION is specified in Table 3.

Table 3 – DEVICE_REVISION attribute

Usage	Attribute	Description
m	integer	Two-octet unsigned integer containing the code for a specific device revision.

7.2.2.3 DEVICE_TYPE

Purpose

The DEVICE_TYPE attribute specifies an identifier for the device type, as defined by the manufacturer of the device. The identifier should be unique for each type.

The device type may specify either a category of devices or a specific product.

Lexical structure

DEVICE_TYPE (integer, identifier)

The attributes of DEVICE_TYPE are specified in Table 4.

Table 4 – DEVICE_TYPE attributes

Usage	Attribute	Description
s	integer	Two-octet unsigned integer containing the code for a specific device type or group of device types.
s	identifier	Reference to a two-octet integer which contains the code for a specific device type or group of device types.
The mapping of the identifier to an integer constant may be stored in an extra file named "devices.cfg". This file contains a list of identifiers for devices or groups of device types.		

7.2.2.4 EDD_PROFILE**Purpose**

The EDD_PROFILE attribute specifies the selected profile of the EDDL as defined in Annex D.

Lexical structure

EDD_PROFILE integer

The attribute of EDD_PROFILE is specified in Table 5.

Table 5 – EDD_PROFILE attribute

Usage	Attribute	Description
o	integer	Two-octet unsigned integer containing the code for the specific profiles selected from Annex D. The following codes are currently defined: 0x01 Profile for HART Communication Foundation (see Clause D.4) 0x02 Profile for Fieldbus Foundation (see Clause D.3) 0x03 Profile for PROFIBUS (see Clause D.2) 0x04 Profile for PROFINET (see Clause D.2)

7.2.2.5 EDD_VERSION**Purpose**

The EDD_VERSION attribute specifies a unique code for the EDD version, which has been used by the manufacturer in constructing the EDD. This attribute shall be included with all device descriptions conforming to this standard.

Lexical structure

EDD_VERSION integer

The attribute of EDD_VERSION is specified in Table 6.

Table 6 – EDD_VERSION attribute

Usage	Attribute	Description
o	integer	Two-octet unsigned integer containing the number 1 for this first edition of the standard.

7.2.2.6 MANUFACTURER

Purpose

The MANUFACTURER attribute identifies the manufacturer. The code allocation should be managed by the specific organizations which are responsible for the different EDDL profiles. The combination of EDD_PROFILE and MANUFACTURER shall be unique.

Lexical structure

MANUFACTURER (long integer, identifier)

The attributes of MANUFACTURER are specified in Table 7.

Table 7 – MANUFACTURER attributes

Usage	Attribute	Description
s	long integer	Unsigned long integer containing the code for a specific manufacturer. If all bits in the value of the integer are set to one, the lexical structure "MANUFACTURER_EXT" shall be present and identifies the manufacturer.
sm	identifier	Reference to an unsigned long integer which contains the code for a specific manufacturer.
The mapping of the identifier to an integer constant should be stored in an extra file named "devices.cfg". This file contains a list of identifiers for manufacturers.		

7.2.2.7 MANUFACTURER_EXT

Purpose

The MANUFACTURER_EXT attribute is reserved for future use. It is a placeholder for a string providing global identification for the MANUFACTURER.

NOTE The specification of this attribute will be included when it becomes internationally available.

Lexical structure

MANUFACTURER_EXT string

The attribute of MANUFACTURER_EXT is specified in Table 8.

Table 8 – MANUFACTURER_EXT attribute

Usage	Attribute	Description
c	string	String containing the information for a unique worldwide manufacturer identification.

7.3 AXIS

7.3.1 General structure

Purpose

AXIS describes an axis of a CHART or GRAPH.

Lexical structure

AXIS identifier [CONSTANT_UNIT , HELP, LABEL, MAX_VALUE, MIN_VALUE, SCALING]

The attributes of AXIS are specified in Table 9.

Table 9 – AXIS attributes

Usage	Attribute	Description
o	CONSTANT_UNIT	Lexical element (see 7.3.2.1).
o	HELP	Lexical element (see 7.3.6.8).
o	LABEL	Lexical element (see 7.3.6.9).
o	MAX_VALUE	Lexical element (see 7.3.2.2).
o	MIN_VALUE	Lexical element (see 7.3.2.3).
o	SCALING	Lexical element (see 7.3.2.4).

7.3.2 Specific attributes

7.3.2.1 CONSTANT_UNIT

Purpose

The CONSTANT_UNIT attribute is used, if an AXIS has a unit code which never changes. The CONSTANT_UNIT is specified as a string that will be displayed along with the value. An AXIS without a constant unit either has no units associated with it or the units are not constant.

Lexical structure

CONSTANT_UNIT (string)<exp>

The attribute of CONSTANT_UNIT is specified in Table 176.

7.3.2.2 MAX_VALUE

Purpose

The MAX_VALUE attribute specifies the value associated with the last tick mark of the AXIS. The last tick mark of an X_AXIS and Y_AXIS is the right-most and top-most tick mark, respectively. By default, the value associated with the last tick mark of an AXIS without a MAX_VALUE attribute is inferred from the WAVEFORMS on the GRAPH.

Lexical structure

MAX_VALUE (reference, double, float, integer, unsigned_integer)<exp>

The attributes of MAX_VALUE are specified in Table 10.

Table 10 – MAX_VALUE, MIN_VALUE attributes

Usage	Attribute	Description
o	reference	Reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance or an element of a REFERENCE_ARRAY instance with a numeric or a date/time data type. MAX_VALUE and MIN_VALUE shall reference VARIABLES of the same data type. This shall apply for X_AXIS and Y_AXIS.
o	double	Double-precision floating-point constant.
o	float	Single-precision floating-point constant.
o	integer	Integer constant.
o	unsigned_integer	Unsigned integer constant.

7.3.2.3 MIN_VALUE

Purpose

The MIN_VALUE attribute specifies the value associated with the first tick mark of the AXIS. The first tick mark of an X_AXIS and Y_AXIS is the left-most and bottom-most tick mark, respectively. By default, the value associated with the first tick mark of an AXIS without a MIN_VALUE attribute is inferred from the WAVEFORMs on the GRAPH.

Lexical structure

MIN_VALUE (reference, double, float, integer, unsigned_integer) <exp>

The attributes of MIN_VALUE are specified in Table 10.

7.3.2.4 SCALING

Purpose

The SCALING attribute specifies the way the tick marks of the AXIS should be scaled. By default, an AXIS without a SCALING attribute is scaled linearly.

Lexical structure

SCALING (LINEAR, LOGARITHMIC) <exp>

The attributes of SCALING are specified in Table 11.

Table 11 – SCALING attributes

Usage	Attribute	Description
s	LINEAR	Tick marks on the axis should be scaled linearly.
s	LOGARITHMIC	Tick marks on the axis should be scaled logarithmically (base 10).

7.4 BLOCK

7.4.1 BLOCK_A

7.4.1.1 General structure

Purpose

A device may be organized in logical blocks. Each BLOCK_A is a logical grouping of PARAMETERS and additional information for BLOCK_A applications. Types of blocks are described by the CHARACTERISTICS attribute. More than one BLOCK_A from the same type can be instantiated.

NOTE This BLOCK_A approach is optimized for access efficiency.

Lexical structure

BLOCK_A identifier [CHARACTERISTICS, LABEL, PARAMETERS, AXIS_ITEMS, CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISPLAY_ITEMS, FILE_ITEMS, GRAPH_ITEMS, GRID_ITEMS, HELP, IMAGE_ITEMS, LIST_ITEMS, MENU_ITEMS, METHOD_ITEMS, PARAMETER_LISTS, PLUGIN_ITEMS, REFERENCE_ARRAY_ITEMS, REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS, WRITE_AS_ONE_ITEMS, CHARTS, FILES, LISTS, GRAPHS, GRIDS, MENUS, METHODS, PLUGINS]

The attributes of BLOCK_A are specified in Table 12.

Table 12 – BLOCK_A attributes

Usage	Attribute	Description
m	CHARACTERISTICS	Lexical element (see 7.4.1.2.1).
m	LABEL	Lexical element (see 7.36.9).
m	PARAMETERS	Lexical element (see 7.4.1.2.2).
o	AXIS_ITEMS	Lexical element (see 7.4.1.2.3).
o	CHART_ITEMS	Lexical element (see 7.4.1.2.4).
o	COLLECTION_ITEMS	Lexical element (see 7.4.1.2.5).
o	EDIT_DISPLAY_ITEMS	Lexical element (see 7.4.1.2.6).
o	FILE_ITEMS	Lexical element (see 7.4.1.2.7).
o	GRAPH_ITEMS	Lexical element (see 7.4.1.2.8).
o	GRID_ITEMS	Lexical element (see 7.4.1.2.9).
o	HELP	Lexical element (see 7.36.8).
o	IMAGE_ITEMS	Lexical element (see 7.4.1.2.10).
o	LIST_ITEMS	Lexical element (see 7.4.1.2.11).
o	LOCAL_PARAMETERS	Lexical element (see 7.4.1.2.12).
o	MENU_ITEMS	Lexical element (see 7.4.1.2.13).
o	METHOD_ITEMS	Lexical element (see 7.4.1.2.14).
o	PARAMETER_LISTS	Lexical element (see 7.4.1.2.15).
o	PLUGIN_ITEMS	Lexical element (see 7.4.1.2.29).
o	REFERENCE_ARRAY_ITEMS	Lexical element (see 7.4.1.2.16).
o	REFRESH_ITEMS	Lexical element (see 7.4.1.2.17).
o	SOURCE_ITEMS	Lexical element (see 7.4.1.2.18).
o	UNIT_ITEMS	Lexical element (see 7.4.1.2.19).
o	WAVEFORM_ITEMS	Lexical element (see 7.4.1.2.20).
o	WRITE_AS_ONE_ITEMS	Lexical element (see 7.4.1.2.21).
o	CHARTS	Lexical element (see 7.4.1.2.22).
o	FILES	Lexical element (see 7.4.1.2.28).
o	LISTS	Lexical element (see 7.4.1.2.23).
o	GRAPHS	Lexical element (see 7.4.1.2.24).
o	GRIDS	Lexical element (see 7.4.1.2.25).
o	MENUS	Lexical element (see 7.4.1.2.26).
o	METHODS	Lexical element (see 7.4.1.2.27).
o	PLUGINS	Lexical element (see 7.4.1.2.30).

7.4.1.2 Specific attributes

7.4.1.2.1 CHARACTERISTICS

Purpose

The CHARACTERISTICS attribute specifies the external characteristics for a BLOCK_A. CHARACTERISTICS may include class, function type, block type and execution time. The external characteristics of a BLOCK_A are contained in a RECORD.

NOTE In general the BLOCK_A CHARACTERISTICS fix the type of the BLOCK_A.

Lexical structure

CHARACTERISTICS reference

The attribute of CHARACTERISTICS is specified in Table 13.

Table 13 – CHARACTERISTIC attribute

Usage	Attribute	Description
m	reference	Reference to a RECORD instance.

7.4.1.2.2 PARAMETERS

Purpose

The PARAMETERS attribute contains a list of references to VARIABLE, VALUE_ARRAY or RECORD instances. Each reference shall have an identifier and may have a description and/or a help text.

NOTE The PARAMETERS in a BLOCK_A correspond to individual communication objects, listed in a device object dictionary. The PARAMETERS elements reference the communication definitions of these objects. The object dictionary maps into the specific location for data in a device.

Lexical structure

PARAMETERS [identifier, reference, description, help]+

The attributes of PARAMETERS are specified in Table 14.

Table 14 – PARAMETER attributes

Usage	Attribute	Description
m	identifier	Identifier of the VARIABLE, VALUE_ARRAY or RECORD. Every element of the PARAMETERS list shall have an identifier to be used in the EDD to refer to it.
m	reference	Reference to a VARIABLE, VALUE_ARRAY or RECORD instance.
o	description	Short description for the reference.
o	help	Help text for the reference.

7.4.1.2.3 AXIS_ITEMS

Purpose

The AXIS_ITEMS attribute is a list of the AXIS instances associated with the BLOCK_A.

Lexical structure

AXIS_ITEMS reference+

The attribute of AXIS_ITEMS is specified in Table 15.

Table 15 – AXIS_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to an AXIS instance.

7.4.1.2.4 CHART_ITEMS

Purpose

The CHART_ITEMS attribute is a list of the CHART instances associated with the BLOCK_A.

Lexical structure

CHART_ITEMS reference+

The attribute of CHART_ITEMS is specified in Table 16.

Table 16 – CHART_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a CHART instance.

7.4.1.2.5 COLLECTION_ITEMS

Purpose

The COLLECTION_ITEMS attribute is a list of the COLLECTION instances associated with the BLOCK_A.

Lexical structure

COLLECTION_ITEMS reference+

The attribute of COLLECTION_ITEMS is specified in Table 17.

Table 17 – COLLECTION_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a COLLECTION instance.

7.4.1.2.6 EDIT_DISPLAY_ITEMS

Purpose

The EDIT_DISPLAY_ITEMS attribute is a list of the EDIT_DISPLAY instances associated with BLOCK_A.

Lexical structure

EDIT_DISPLAY_ITEMS reference+

The attribute of EDIT_DISPLAY_ITEMS is specified in Table 18.

Table 18 – EDIT_DISPLAY_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to an EDIT_DISPLAY instance.

7.4.1.2.7 FILE_ITEMS**Purpose**

The FILE_ITEMS attribute is a list of the FILE instances associated with the BLOCK_A.

Lexical structure

FILE_ITEMS reference+

The attribute of FILE_ITEMS is specified in Table 19.

Table 19 – FILE_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a FILE instance.

7.4.1.2.8 GRAPH_ITEMS**Purpose**

The GRAPH_ITEMS attribute is a list of the GRAPH instances associated with the BLOCK_A.

Lexical structure

GRAPH_ITEMS reference+

The attribute of GRAPH_ITEMS is specified in Table 20.

Table 20 – GRAPH_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a GRAPH instance.

7.4.1.2.9 GRID_ITEMS**Purpose**

The GRID_ITEMS attribute is a list of the GRID instances associated with the BLOCK_A.

Lexical structure

GRID_ITEMS reference+

The attribute of GRID_ITEMS is specified in Table 21.

Table 21 – GRID_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a GRID instance.

7.4.1.2.10 IMAGE_ITEMS**Purpose**

The IMAGE_ITEMS attribute is a list of the IMAGE instances associated with BLOCK_A.

Lexical structure

IMAGE_ITEMS reference+

The attribute of IMAGE_ITEMS is specified in Table 22.

Table 22 – IMAGE_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to an IMAGE instance.

7.4.1.2.11 LIST_ITEMS

Purpose

The LIST_ITEMS attribute is a list of the LIST instances associated with BLOCK_A.

Lexical structure

LIST_ITEMS reference+

The attribute of LIST_ITEMS is specified in Table 23.

Table 23 – LIST_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a LIST instance.

7.4.1.2.12 LOCAL_PARAMETERS

Purpose

The LOCAL_PARAMETERS attribute contains a list of references to VARIABLE, VALUE_ARRAY or RECORD instances. Each reference shall have an identifier and may have a description and/or a help text.

NOTE The LOCAL_PARAMETERS in a BLOCK_A do not correspond to communication objects; they are parameters that are only used by the EDD application.

Lexical structure

LOCAL_PARAMETERS [identifier, reference, description, help] +

The attributes of LOCAL_PARAMETERS are specified in Table 14.

7.4.1.2.13 MENU_ITEMS

Purpose

The MENU_ITEMS attribute specifies the MENUs associated with BLOCK_A.

Lexical structure

MENU_ITEMS reference+

The attribute of MENU_ITEMS is specified in Table 24.

Table 24 – MENU_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a MENU.

7.4.1.2.14 METHOD_ITEMS**Purpose**

The METHOD_ITEMS attribute specifies the METHODS associated with BLOCK_A.

Lexical structure

METHOD_ITEMS reference+

The attribute of METHOD_ITEMS is specified in Table 25.

Table 25 – METHOD_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a METHOD instance.

7.4.1.2.15 PARAMETER_LISTS**Purpose**

The PARAMETER_LISTS attribute specifies the VARIABLE_LISTs associated with a BLOCK_A, plus an optional description and help for each VARIABLE_LIST. Every BLOCK_A may contain multiple PARAMETER_LISTS.

Lexical structure

PARAMETER_LISTS [identifier, reference, description, help] +

The attributes of PARAMETER_LISTS are specified in Table 26.

Table 26 – PARAMETER_LISTS attributes

Usage	Attribute	Description
m	identifier	Identifier of the VARIABLE_LIST. Every element of the PARAMETER_LIST list shall have an identifier to be used in the EDD to refer to it.
m	reference	Reference to a VARIABLE_LIST instance.
o	description	Short description of the item.
o	help	Help text for the item.

7.4.1.2.16 REFERENCE_ARRAY_ITEMS**Purpose**

The REFERENCE_ARRAY_ITEMS attribute specifies the REFERENCE_ARRAYs associated with the BLOCK_A.

Lexical structure

REFERENCE_ARRAY_ITEMS reference+

The attribute of REFERENCE_ARRAY_ITEMS is specified in Table 27.

Table 27 – REFERENCE_ARRAY_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a REFERENCE_ARRAY instance.

7.4.1.2.17 REFRESH_ITEMS

Purpose

The REFRESH_ITEMS attribute specifies the REFRESH relations associated with the BLOCK_A.

Lexical structure

REFRESH_ITEMS reference+

The attribute of REFRESH_ITEMS is specified in Table 28.

Table 28 – REFRESH_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a REFRESH instance.

7.4.1.2.18 SOURCE_ITEMS

Purpose

The SOURCE_ITEMS attribute specifies the SOURCES associated with the BLOCK_A.

Lexical structure

SOURCE_ITEMS reference+

The attribute of SOURCE_ITEMS is specified in Table 29.

Table 29 – SOURCE_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a SOURCE instance.

7.4.1.2.19 UNIT_ITEMS

Purpose

The UNIT_ITEMS attribute specifies the UNIT relations associated with the BLOCK_A.

Lexical structure

UNIT_ITEMS reference+

The attribute of UNIT_ITEMS is specified in Table 30.

Table 30 – UNIT_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to an UNIT instance.

7.4.1.2.20 WAVEFORM_ITEMS**Purpose**

The WAVEFORM_ITEMS attribute specifies the WAVEFORMs associated with the BLOCK_A.

Lexical structure

WAVEFORM_ITEMS reference+

The attribute of WAVEFORM_ITEMS is specified in Table 31.

Table 31 – WAVEFORM_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a WAVEFORM instance.

7.4.1.2.21 WRITE_AS_ONE_ITEMS**Purpose**

The WRITE_AS_ONE_ITEMS attribute specifies the WRITE_AS_ONE relations associated with the BLOCK_A.

Lexical structure

WRITE_AS_ONE_ITEMS reference+

The attribute of WRITE_AS_ONE_ITEMS is specified in Table 32.

Table 32 – WRITE_AS_ONE_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a WRITE_AS_ONE instance.

7.4.1.2.22 CHARTS**Purpose**

The CHARTS attribute specifies the CHARTs associated with the BLOCK_A. Only those CHARTs that will be referenced via a specific BLOCK_A instance (see 7.38.21) shall be specified.

Lexical structure

CHARTS (identifier, reference, description, help)+

The attributes of CHARTS are specified in Table 33.

Table 33 – CHARTS attributes

Usage	Attribute	Description
m	identifier	Name by which the CHART may be referenced.
m	reference	Reference to a CHART instance.
o	description	Short description of the CHART.
o	help	Help text for the CHART.

7.4.1.2.23 LISTS**Purpose**

The LISTS attribute specifies the LISTS associated with the BLOCK_A. Only those LISTS that will be referenced via a specific BLOCK_A instance (see 7.4.1) shall be specified.

Lexical structure

LISTS (identifier, reference, description, help)+

The attributes of LISTS are specified in Table 34.

Table 34 – LISTS attributes

Usage	Attribute	Description
m	identifier	Name by which the LIST may be referenced.
m	reference	Reference to a LIST instance.
o	description	Short description of the LIST.
o	help	Help text for the LIST.

7.4.1.2.24 GRAPHS**Purpose**

The GRAPHS attribute specifies the GRAPHS associated with the BLOCK_A. Only those GRAPHS that will be referenced via a specific BLOCK_A instance (see 7.38.23) shall be specified.

Lexical structure

GRAPHS (identifier, reference, description, help)+

The attributes of GRAPHS are specified in Table 35.

Table 35 – GRAPHS attributes

Usage	Attribute	Description
m	identifier	Name by which the GRAPH may be referenced.
m	reference	Reference to a GRAPH instance.
o	description	Short description of the GRAPH.
o	help	Help text for the GRAPH.

7.4.1.2.25 GRIDS

Purpose

The GRIDS attribute specifies the GRIDS associated with the BLOCK_A. Only those GRIDS that will be referenced via a specific BLOCK_A instance (see 7.38.24) shall be specified.

Lexical structure

GRIDS (identifier, reference, description, help)+

The attributes of GRIDS are specified in Table 36.

Table 36 – GRIDS attributes

Usage	Attribute	Description
m	identifier	Name by which the GRID may be referenced.
m	reference	Reference to a GRID instance.
o	description	Short description of the GRID.
o	help	Help text for the GRID.

7.4.1.2.26 MENUS

Purpose

The MENUS attribute specifies the MENUS associated with the BLOCK_A. Only those MENUS that will be referenced via a specific BLOCK_A instance (see 7.38.25) shall be specified.

Lexical structure

MENUS (identifier, reference, description, help)+

The attributes of MENUS are specified in Table 37.

Table 37 – MENUS attributes

Usage	Attribute	Description
m	identifier	Name by which the MENU may be referenced.
m	reference	Reference to a MENU instance.
o	description	Short description of the MENU.
o	help	Help text for the MENU.

7.4.1.2.27 METHODS

Purpose

The METHODS attribute specifies the METHODS associated with the BLOCK_A. Only those METHODS that will be referenced via a specific BLOCK_A instance (see 7.38.26) shall be specified.

Lexical structure

METHODS (identifier, reference, description, help)+

The attributes of METHODS are specified in Table 38.

Table 38 – METHODS attributes

Usage	Attribute	Description
m	identifier	Name by which the METHOD may be referenced.
m	reference	Reference to a METHOD instance.
o	description	Short description of the METHOD.
o	help	Help text for the METHOD.

7.4.1.2.28 FILES**Purpose**

The FILES attribute specifies the FILES associated with the BLOCK_A. Only those FILES that will be referenced via a specific BLOCK_A instance (see 7.38.29) shall be specified.

Lexical structure

FILES (identifier, reference, description, help)+

The attributes of FILES are specified in Table 39.

Table 39 – FILES attributes

Usage	Attribute	Description
m	identifier	Name by which the FILE may be referenced.
m	reference	Reference to a FILE instance.
o	description	Short description of the FILE.
o	help	Help text for the FILE.

7.4.1.2.29 PLUGIN_ITEMS**Purpose**

The PLUGIN_ITEMS attribute specifies the PLUGINS associated with BLOCK_A.

Lexical structure

PLUGIN_ITEMS reference+

The attribute of PLUGIN_ITEMS is specified in Table 40.

Table 40 – PLUGIN_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a PLUGIN instance.

7.4.1.2.30 PLUGINS**Purpose**

The PLUGINS attribute specifies the PLUGINS associated with the BLOCK_A. Only those PLUGINS that will be referenced via a specific BLOCK_A instance (see 7.38.30) shall be specified.

Lexical structure

PLUGINS (identifier, reference, description, help)+

The attributes of PLUGINS are specified in Table 41.

Table 41 – PLUGINS attributes

Usage	Attribute	Description
m	identifier	Name by which the PLUGIN may be referenced.
m	reference	Reference to a PLUGIN instance.
o	description	Short description of the PLUGIN.
o	help	Help text for the PLUGIN.

7.4.2 BLOCK_B

7.4.2.1 General structure

Purpose

The variables of a device or module respectively are structured in blocks corresponding to device components or functional parts. Three types of blocks are defined: PHYSICAL, TRANSDUCER and FUNCTION. More than one BLOCK_B from the same TYPE can be instantiated. For efficient access, each instance has its own NUMBER. For each of the block types there are different threads of NUMBER sequences, starting with 1.

When accessing a VARIABLE within a device, the BLOCK_B construct is used in conjunction with the COMMAND to provide relative addressing.

NOTE 1 This BLOCK_B approach is optimized for memory efficiency.

NOTE 2 The storage of a variable in a device is manufacturer specific and is represented by a device directory. A device directory contains a summary of the available BLOCK_B entries in a device. Furthermore, the device directory contains for each BLOCK_B a pointer to its physical address. An EDD application finds a single object by adding an offset to the BLOCK_B physical address.

NOTE 3 Within a BLOCK_B instance, relative indexing is used. This relative index is defined in the device profile.

Lexical structure

BLOCK_B identifier [NUMBER, TYPE]

The attributes of BLOCK_B are specified in Table 42.

Table 42 – BLOCK_B attributes

Usage	Attribute	Description
m	NUMBER	Lexical element (see 7.4.2.2.1).
m	TYPE	Lexical element (see 7.4.2.2.2).

7.4.2.2 Specific attributes

7.4.2.2.1 NUMBER

Purpose

The NUMBER attribute enumerates each BLOCK_B of the same TYPE in the device by an integer greater than zero. A device may contain several BLOCK_B instances, which are identified with the NUMBER attribute.

Lexical structure

NUMBER (integer, expression)<exp>

The attributes of NUMBER are specified in Table 43.

Table 43 – NUMBER attributes

Usage	Attribute	Description
s	integer	Number of the BLOCK_B instance of the same TYPE.
s	expression	Expression which shall be evaluated to a non-negative integer that is within the index range of the NUMBER attribute, for the description of expressions, see 7.40.

7.4.2.2.2 TYPE

Purpose

The TYPE attribute is used to specify one of the three block types.

Lexical structure

TYPE [PHYSICAL, TRANSDUCER, FUNCTION]

The attributes of TYPE are specified in Table 44.

Table 44 – TYPE attributes

Usage	Attribute	Description
s	FUNCTION	Named block consisting of one or more input, output and contained parameters. Function blocks represent the basic automation functions performed by an EDD application, which is independent of the specific devices and the network. Each function block processes input parameters according to a specified algorithm and an internal set of contained parameters. They produce output parameters that are available for use within the same function block EDD application or by other function block applications.
s	PHYSICAL	Hardware/resource specific characteristics of a device are made visible through the physical block. Similar to transducer blocks, they isolate function blocks from the physical hardware by containing a set of implementation independent hardware/resource parameters.
s	TRANSDUCER	Transducer blocks isolate function blocks from the specifics of I/O devices, such as sensors, actuators, and switches. Transducer blocks control access to I/O devices through a device-independent interface defined for use by function blocks. Transducer blocks perform functions, such as calibration and linearization, on I/O data to convert it to a device independent representation.

7.5 CHART

7.5.1 General structure

Purpose

CHART describes a chart used to display data from a device in the EDD application. A CHART is used to display continuous data values from the device, as opposed to a GRAPH that is used to display a data set that is stored in a device. The data from the device may be adjusted prior to being displayed via the use of INIT_ACTIONS and RERESH_ACTIONS on the SOURCE members of the CHART.

Lexical structure

CHART identifier [MEMBERS, CYCLE_TIME, HEIGHT, WIDTH, LENGTH, HELP, LABEL, TYPE, VALIDITY, VISIBILITY]

The attributes of CHART are specified in Table 45.

Table 45 – CHART attributes

Usage	Attribute	Description
M	MEMBERS	Lexical element (see 7.36.12).
o	CYCLE_TIME	Lexical element (see 7.5.2.1).
o	HEIGHT	Lexical element (see 7.36.7).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	LENGTH	Lexical element (see 7.5.2.2).
o	TYPE	Lexical element (see 7.5.2.3).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	WIDTH	Lexical element (see 7.36.17).

7.5.2 Specific attributes

7.5.2.1 CYCLE_TIME

Purpose

The CYCLE_TIME attribute specifies the length of time, in ms, the EDD application shall wait between successive reads of the source values in the device. By default, the cycle time of a CHART without a CYCLE_TIME attribute is 1 000 ms.

Lexical structure

CYCLE_TIME (integer)<exp>

The attribute of CYCLE_TIME is specified in Table 46.

Table 46 – CYCLE_TIME attribute

Usage	Attribute	Description
m	integer	Length of time, in ms, between successive reads of the source values in the device. After the CHART is displayed by an EDD application, the CYCLE_TIME shall not change.

7.5.2.2 LENGTH**Purpose**

The LENGTH attribute specifies the length of time, in ms, that is displayed by the CHART. The number of samples displayed by a chart can be calculated by dividing the LENGTH attribute by the CYCLE_TIME attribute. By default, the cycle time of a CHART without a LENGTH attribute is 600 000 ms.

Lexical structure

LENGTH (integer)<exp>

The attribute of LENGTH is specified in Table 47.

Table 47 – LENGTH attribute

Usage	Attribute	Description
m	integer	Length of time, in ms, that is displayed by the CHART. After the CHART is displayed by an EDD application, the LENGTH shall not change.

7.5.2.3 TYPE**Purpose**

The TYPE attribute specifies the type of CHART the EDD shall display. By default, the type of a CHART without a TYPE attribute is STRIP.

Lexical structure

TYPE (GAUGE, HORIZONTAL_BAR, SCOPE, STRIP, SWEEP, VERTICAL_BAR)<exp>

The attributes of TYPE are specified in Table 48.

Table 48 – TYPE attributes

Usage	Attribute	Description
s	GAUGE	A single source value is displayed as a gauge, which is a graphical representation of the data similar to the fuel gauge in an automobile.
s	HORIZONTAL_BAR	The source values are displayed as a horizontal bar chart.
s	SCOPE	The data in this diagram is updated from left to right. When the source values reach the end of the display area of the CHART the display area is erased and the new source values are once again displayed starting at the beginning of the display area. The X-axis shall be updated appropriately to the display area.
s	STRIP	The data in this diagram is updated from left to right. When the source values reach the end of the display area of the CHART, the display area is scrolled with the oldest source values being removed from the display area and the newest source values being added to the display area. The X-axis shall be scrolled and updated appropriately to the display area.
s	SWEEP	The data in this diagram is updated from left to right. When the source values reach the end of the display area of the CHART, the new source values are displayed starting at the beginning of the display area, but unlike SCOPE only the portion of the display area needed to display the new source values is erased. The X-axis shall be scrolled and updated appropriately to the display area.
s	VERTICAL_BAR	The source values are displayed as a vertical bar chart.

7.6 COLLECTION

7.6.1 General structure

Purpose

A COLLECTION is a logical group of items, such as VARIABLES or MENUs. An identifier is assigned to each item. The items may be referenced within the device description by using the COLLECTION identifier and the item name.

COLLECTION is intended to be used, for example to identify parameters that should be processed together. The EDD application should at least support nesting of two levels, for example COLLECTION of COLLECTION.

Lexical structure

COLLECTION identifier, item-type, [MEMBERS, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]

The attributes of COLLECTION are specified in Table 49.

Table 49 – COLLECTION attributes

Usage	Attribute	Description
o	item-type	Lexical element (see 7.6.2).
m	MEMBERS	Lexical element (see 7.36.12).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	PRIVATE	Lexical element (see 7.36.18).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).

7.6.2 Specific attribute – item-type

Purpose

The item-type attribute specifies the type of members in the collection. If item-type is specified, all the collection members shall be of the specified type. Otherwise, the collection members may be of any type.

Lexical structure

item-type

Table 50 shows the allowed item-types.

Table 50 – item-type

Usage	item-type	Description
s	BLOCK_A	Selection of this basic construct.
s	BLOCK_B	Selection of this basic construct.
s	CHART	Selection of this basic construct.
s	COLLECTION	Selection of this basic construct.
s	EDIT_DISPLAY	Selection of this basic construct.
s	FILE	Selection of this basic construct.
s	GRAPH	Selection of this basic construct.
s	GRID	Selection of this basic construct.
s	IMAGE	Selection of this basic construct.
s	LIST	Selection of this basic construct.
s	MENU	Selection of this basic construct.
s	METHOD	Selection of this basic construct.
s	PLUGIN	Selection of this basic construct.
s	RECORD	Selection of this basic construct.
s	REFERENCE_ARRAY	Selection of this basic construct.
s	VALUE_ARRAY	Selection of this basic construct.
s	VARIABLE	Selection of this basic construct.
s	VARIABLE_LIST	Selection of this basic construct.

7.7 COMMAND

7.7.1 General structure

Purpose

COMMAND is a construct to support mapping of the EDD communication elements to a chosen communication system. It specifies various elements required to build a communication frame. If this construct is used by an EDDL profile, every data item needing to be transmitted shall be referenced inside a COMMAND and also in the upload/download MENU.

NOTE 1 The address field of a communication frame is specified using the selectable constructs BLOCK, SLOT, SUB_SLOT, NUMBER, INDEX as applicable. The type of the communication frame or its control field is specified by the OPERATION construct. The data content of the communication frame is specified by the TRANSACTION construct.

Lexical structure

COMMAND identifier, [OPERATION, TRANSACTION+, INDEX, BLOCK_B, NUMBER, SLOT, SUB_SLOT, API, HEADER, POST_RECEIVE_ACTIONS, RESPONSE_CODES]

The attributes of COMMAND are specified in Table 51.

Table 51 – COMMAND attributes

Usage	Attribute	Description
m	OPERATION	Lexical element (see 7.7.2.1).
m	TRANSACTION	Lexical element (see 7.7.2.2).
c	INDEX	Lexical element (see 7.7.2.3).
s	BLOCK_B	Lexical element (see 7.7.2.4).
s	NUMBER	Lexical element (see 7.7.2.5).
s	SLOT	Lexical element (see 7.7.2.6).
s	SUB_SLOT	Lexical element (see 7.7.2.7).
o	API	Lexical element (see 7.7.2.11).
o	HEADER	Lexical element (see 7.7.2.9).
o	POST_RQSTRECEIVE_ACTIONS	Lexical element (see 7.7.2.12).
o	RESPONSE_CODES	Lexical element (see 7.36.14).

NOTE 2 Response codes appearing within a transaction apply only to that transaction.

NOTE 3 Response codes appearing outside of any transaction apply to all transactions.

NOTE 4 If the same response code is specified both inside and outside of a transaction, the response code specified within the transaction takes precedence, but only for that transaction.

7.7.2 Specific attributes

7.7.2.1 OPERATION

Purpose

The OPERATION attribute specifies the actions taken by the device upon receiving a service request from the communication system.

Lexical structure

OPERATION [(COMMAND, READ, WRITE)]

The attributes of OPERATION are specified in Table 52.

Table 52 – OPERATION attributes

Usage	Attribute	Description
s	COMMAND	The device performs a predefined set of actions, for example self-test, master reset.
s	READ	Specifies a read transaction. The device returns the current values of the specified variables. VARIABLES shall be defined in the REPLY section of TRANSACTION.
s	WRITE	Specifies a write transaction. The device receives the current values of the specified variables. VARIABLES shall be defined in the REQUEST section of TRANSACTION.

7.7.2.2 TRANSACTION

7.7.2.2.1 General structure

Purpose

Transactions specify the data field of the command's request and reply messages. It is possible to define more than one transaction. The syntax should support a unique identification of the TRANSACTION. Each TRANSACTION may include a set of response codes that apply only to that particular transaction.

EXAMPLE Block commands, such as commands 4 and 5 of HART⁶ Universal Command Revision 4 or later, are examples of multiple transaction commands.

Lexical structure

TRANSACTION [REPLY, REQUEST, integer, RESPONSE_CODES]

The attributes of TRANSACTION are specified in Table 53.

Table 53 – TRANSACTION attributes

Usage	Attribute	Description
m	REPLY	Lexical element (see 7.7.2.2.2.1).
m	REQUEST	Lexical element (see 7.7.2.2.2.2).
o	integer	Number of the TRANSACTIONS, if more than one TRANSACTION is used.
o	RESPONSE_CODES	Lexical element (see 7.36.14).

7.7.2.2.2 Specific attributes

7.7.2.2.2.1 REPLY

Purpose

The REPLY attribute contains a list of data items (constants or references to variables, lists, arrays, or collections), which are received from the device. The order of the list shall be maintained in the corresponding data field in the communicated PDU. If the referenced data items are not of type VARIABLE, the referenced data inside the EDD elements shall always terminate as references of VARIABLE. All other referenced EDD element types are not allowed. The combination of list, array, and collection with an item-mask is not allowed.

When a VARIABLE does not begin and end on octet boundaries, a mask should be used to specify how the variable is packed into the data field.

Every bit in the mask may have a semantical meaning. In addition, if the least significant bit is set in an item-mask, the next data item (if any) is contained in the following octet(s) of the PDU. If the least significant bit is clear (not set), the next data item is contained in the same octet(s) of the PDU.

When a LIST is referenced, the EDD application shall reset the LIST and after that it shall fill the LIST with the data of the received PDU.

Even if the least significant bit is not used for actual data, it should be set in an item-mask (associated with a "dummy" data item) if data items follow.

⁶ See Annex D.4.

Lexical structure

REPLY [integer, reference [item-mask, INFO, INDEX]]+

The attributes of REPLY are specified in Table 54.

Table 54 – REPLY and REQUEST attributes

Usage	Attribute	Description
s	integer	Integer literal which can have different sizes.
s	reference	Reference to a VARIABLE, ARRAY, COLLECTION, REFERENCE_ARRAY, or LIST instance.
o	item-mask	Bit pattern to extract the VARIABLE from the data field. The item-mask shall be used only for the data types INTEGER, UNSIGNED INTEGER, ENUMERATED and BIT_ENUMERATED. The length of the item-mask shall be 1 byte to 4 bytes and shall correspond to the length of the data item in the REQUEST or REPLY list.
o	INFO	The VARIABLE is not actually stored in the device. It provides additional information to ensure correct processing.
o	INDEX	The VARIABLE is used in the REQUEST or REPLY as an index into a REFERENCE_ARRAY.

A variable may be qualified with both INDEX and INFO, in which case it is called a local index variable. Commands with local indices behave exactly the same as commands with INDEX qualified variables except that the index variables are not stored in the device.

INFO may be used to send or read a VARIABLE with units differing from those used in the device.

7.7.2.2.2.2 REQUEST

Purpose

The REQUEST attribute contains a list of data items (constants or references to variables, lists, arrays, or collections), which are sent to the device. The order of the list shall be maintained in the corresponding data field in the communicated PDU. If the referenced data items are not of type VARIABLE, the referenced data inside the EDD elements shall always terminate as references of VARIABLE. All other referenced EDD element types are not allowed. The combination of list, array, and collection with an item-mask is not allowed.

When a data item does not begin and end on octet boundaries of the PDU, an item-mask shall be used to specify how the data item is packed into the data field. The item-mask is an integer number of octets that shall be logically AND-ed with the communicated PDU to extract the desired data item.

If no data item-mask is specified, an implicit mask is used with all bits set for every octet, corresponding to the data item length.

Every bit in the mask may have a semantic meaning. In addition, if the least significant bit is set in an item-mask, the next data item (if any) is contained in the following octet(s) of the PDU. If the least significant bit is clear (not set), the next data item is contained in the same octet(s) of the PDU.

Even if the least significant bit is not used for actual data it should be set in an item-mask (associated with a “dummy” data item) if data items follow.

Lexical structure

REQUEST [integer, reference [item-mask, INFO, INDEX]]+

The attributes of REQUEST are specified in Table 54.

7.7.2.3 INDEX

Purpose

The INDEX attribute shall be present if the SLOT or BLOCK_B attribute is used. The INDEX specifies the data item address within the SLOT or the BLOCK_B.

Lexical structure

INDEX (integer, reference, expression)<exp>

The attributes of INDEX are specified in Table 55.

Table 55 – INDEX attributes

Usage	Attribute	Description
s	integer	Number of the index.
s	reference	Reference to a VARIABLE instance containing the index.
s	expression	Expression which shall be evaluated to a non-negative integer that is within the index range of the INDEX attribute; for the description of expressions, see 7.40.

7.7.2.4 BLOCK_B

Purpose

The BLOCK_B attribute provides a reference to an individual block within the device.

Lexical structure

BLOCK_B (reference)<exp>

The attribute of BLOCK_B is specified in Table 56.

Table 56 – BLOCK_B attribute

Usage	Attribute	Description
m	reference	Reference to a BLOCK_B instance.

7.7.2.5 NUMBER

Purpose

The NUMBER attribute is used to enumerate the COMMANDS.

Lexical structure

NUMBER unsigned_integer

The attribute of NUMBER is specified in Table 57.

Table 57 – NUMBER attribute

Usage	Attribute	Description
m	unsigned_integer	Number of the COMMAND.

7.7.2.6 SLOT**Purpose**

The SLOT attribute specifies the number of a slot. Data items of a device may be allocated to logical slots. Within a slot each data item is addressed using an index.

Lexical structure

SLOT (integer, reference, expression)<exp>

The attributes of SLOT are specified in Table 58.

Table 58 – SLOT attributes

Usage	Attribute	Description
s	integer	Number of the slot.
s	reference	Reference to a VARIABLE instance containing the slot number.
s	expression	Expression, which shall be evaluated to a non-negative integer that is within the index range of the SLOT attribute; for the description of expressions, see 7.40.

7.7.2.7 SUB_SLOT**Purpose**

The SUB_SLOT attribute specifies the number of a sub-slot. Data items of a device may be allocated to logical sub-slots. Within a sub-slot each data item is addressed using an index.

Lexical structure

SUB_SLOT (integer, reference, expression)<exp>

The attributes of SUB_SLOT are specified in Table 59.

Table 59 – SUB_SLOT attributes

Usage	Attribute	Description
s	integer	Number of the sub-slot.
s	reference	Reference to a VARIABLE instance containing the sub-slot number.
s	expression	Expression, which shall be evaluated to a non-negative integer that is within the index range of the SUB_SLOT attribute; for the description of expressions, see 7.40.

7.7.2.8 CONNECTION (void)**7.7.2.9 HEADER****Purpose**

The HEADER attribute is provided for specific communication protocols.

It may be used for addressing data in a device or for a reset operation.

Lexical structure

HEADER (string)<exp>

The attribute of HEADER is specified in Table 60.

Table 60 – HEADER attribute

Usage	Attribute	Description
m	string	String that is used for the communication with specific communication protocols.

7.7.2.10 MODULE (void)

7.7.2.11 API

Purpose

The API attribute (Application Process Identifier) specifies the user profile and is used as an additional addressing information. By default, API is 0 for unrestricted use.

Lexical structure

API (integer, reference, expression)<exp>

The attributes of API are specified in Table 61.

Table 61 – API attributes

Usage	Attribute	Description
s	integer	Number of the API.
s	reference	Reference to a VARIABLE instance containing the API number.
s	expression	Expression, which shall be evaluated to a non-negative integer for the API number; for the description of expressions, see 7.40.

7.7.2.12 POST_RQSTRECEIVE_ACTIONS

Purpose

POST_RQSTRECEIVE_ACTIONS shall only be supported by device simulators.

The POST_RQSTUPDATE_ACTIONS attribute specifies METHODS that shall be executed by the device simulator after all of the values in the REQUEST section have been updated and their POST_RQSTUPDATE_ACTIONS, if defined, run and before the REPLY section is generated. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

Lexical structure

POST_RQSTRECEIVE_ACTIONS [(reference)<exp>]+

The attribute of POST_RQSTRECEIVE_ACTIONS is specified in Table 62.

Table 62 – POST_RQSTRECEIVE_ACTIONS attribute

Usage	Attribute	Description
m	reference	Reference to a METHOD.

7.8 COMPONENT

7.8.1 General structure

Purpose

Describes the configuration behavior of the device that is described in the EDD. A component enumerates all relations to other components.

A component consists of one DDL file, import files, include files, dictionaries, images, and help files.

All modules of a modular device shall be described with COMPONENTs. The relations between the different components are described with COMPONENT_RELATIONS. The hierarchy of the catalog is described with COMPONENT_FOLDERS and COMPONENTs. Conditional expressions can have access to different components using OBJECT_REFERENCES. OBJECT_REFERENCES allow access to any variables and allow to call methods.

Lexical structure

COMPONENT identifier [LABEL, BYTE_ORDER, CAN_DELETE, CHECK_CONFIGURATION, CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, COMPONENT_RELATIONS, CONNECTION_POINT, DECLARATION, DETECT, EDD, HELP, INITIAL_VALUES, PRODUCT_URI, PROTOCOL, REDUNDANCY, SCAN, SCAN_LIST, SUPPLIED_INTERFACE]

The attributes are specified in Table 63.

Table 63 – COMPONENT attributes

Usage	Attribute	Description
m	LABEL	Lexical element (see 7.36.9).
o	BYTE_ORDER	Lexical element (see 7.8.2.11).
o	CAN_DELETE	Lexical element (see 7.8.2.1).
o	CHECK_CONFIGURATION	Lexical element (see 7.8.2.2).
o	CLASSIFICATION	Lexical element (see 7.36.1).
o	COMPONENT_PARENT	Lexical element (see 7.36.2).
o	COMPONENT_PATH	Lexical element (see 7.36.3).
o	COMPONENT_RELATIONS	Lexical element (see 7.8.2.3).
o	CONNECTION_POINT	Lexical element (see 7.8.2.12).
o	DECLARATION	Lexical element (see 7.8.2.4).
o	DETECT	Lexical element (see 7.8.2.5).
o	EDD	Lexical element (see 7.8.2.6).
o	HELP	Lexical element (see 7.36.8).
o	INITIAL_VALUES	Lexical element (see 7.8.2.7).
o	PRODUCT_URI	Lexical element (see 7.8.2.13).
o	PROTOCOL	Lexical element (see 7.36.13).
o	REDUNDANCY	Lexical element (see 7.8.2.8).
o	SCAN	Lexical element (see 7.8.2.9).
o	SCAN_LIST	Lexical element (see 7.8.2.10).
o	SUPPLIED_INTERFACE	Lexical element (see 7.36.15).

7.8.2 Specific attributes

7.8.2.1 CAN_DELETE

Purpose

The CAN_DELETE attribute resolves to a TRUE or FALSE that indicates whether it is safe to delete this module instance from a modular device. This attribute is used in online and offline use cases. By default, CAN_DELETE is TRUE. An example thereof is: channels of a module cannot be deleted; only the module as a whole can be deleted.

Lexical structure

CAN_DELETE (FALSE, TRUE)<exp>

The attributes of CAN_DELETE are specified in Table 64.

Table 64 – CAN_DELETE attributes

Usage	Attribute	Description
s	TRUE	Component can be deleted.
s	FALSE	Component cannot be deleted.

7.8.2.2 CHECK_CONFIGURATION

Purpose

The CHECK_CONFIGURATION attribute references a method that shall be called by the EDD application to confirm the current configuration of this COMPONENT. A BI_SUCCESS return value indicates that the configuration is valid; all other values indicate that the configuration is invalid. If the method aborts, then the configuration is invalid. The referenced method shall have no input parameters and a single output parameter of type DD_STRING that returns a message. This message shall be displayed to the user by the EDD application. This message may contain several reasons for an invalid configuration. In this case the message can contain carriage return and line feed. This method may be used for both online and offline.

Lexical structure

CHECK_CONFIGURATION method_reference

The attribute of CHECK_CONFIGURATION is specified in Table 65.

Table 65 – CHECK_CONFIGURATION attribute

Usage	Attribute	Description
m	method_reference	Reference to a METHOD.

7.8.2.3 COMPONENT_RELATIONS

Purpose

The COMPONENT_RELATIONS attribute lists the relations to other components that this modular device or module can support. This EDD can list sibling-modules, child-modules and parent-modules.

Lexical structure

COMPONENT_RELATIONS [(component-relation-reference)<exp>]+

The attribute of COMPONENT_RELATIONS is specified in Table 66.

Table 66 – COMPONENT_RELATIONS attribute

Usage	Attribute	Description
m	component-relation-reference	List of component relation identifiers. Conditional expressions are allowed.

7.8.2.4 DECLARATION

Purpose

The DECLARATION attribute specifies basic constructs associated with the component.

Lexical structure

DECLARATION [item]+

The attribute of DECLARATION is specified in Table 67.

Table 67 – DECLARATION attribute

Usage	Attribute	Description
m	item	Declarations of any basic constructs (see 7.1.4).

7.8.2.5 DETECT

Purpose

The DETECT attribute indicates the method that should be called by the EDD application to detect a component.

In HART the EDD application should take care to read with command 75 to identify a component.

Lexical structure

DETECT method_reference

The attribute of DETECT is specified in Table 68.

Table 68 – DETECT attribute

Usage	Attribute	Description
s	method_reference	Reference to a detect method.

7.8.2.6 EDD

Purpose

The EDD attribute specifies a file that contains the rest of the EDD. This is only required if the component description is in a separate file.

Lexical structure

EDD string

The attribute of EDD is specified in Table 69.

Table 69 – EDD attribute

Usage	Attribute	Description
m	string	File that contains the rest of the EDD.

7.8.2.7 INITIAL_VALUES

Purpose

The INITIAL_VALUES attribute is a set of initialization values for VARIABLES, VALUE_ARRAYs or LISTs. Even if the VARIABLES do have an INITIAL_VALUE or DEFAULT_VALUE, the INITIAL_VALUES of the COMPONENT shall be used for initialization.

It could be used for example to define a different initialization for different usages such as differential pressure, level and flow measurement for a pressure transmitter.

Lexical structure

INITIAL_VALUES [reference, [(value<exp>)*, ((value<exp>)+)*]]+

The attributes of INITIAL_VALUES are specified in Table 65.

Table 70 – INITIAL_VALUES attributes

Usage	Attribute	Description
s	reference	Reference to a VARIABLE, VALUE_ARRAY or LIST.
s	value	Value to initialize the referenced VARIABLE. For a VALUE_ARRAY or a LIST, a set of values. May be nested for a multidimensional VALUE_ARRAY or LIST.

7.8.2.8 REDUNDANCY**Purpose**

The REDUNDANCY attribute resolves to an integer (defaults to one) that indicates how many redundant sockets are supported by this device.

Lexical structure

REDUNDANCY (integer)<exp>

The attribute of REDUNDANCY is specified in Table 71.

Table 71 – REDUNDANCY attribute

Usage	Attribute	Description
o	integer	Number of connection points.

7.8.2.9 SCAN**Purpose**

The SCAN attribute indicates the method that should be called by the EDD application to create a life list of a modular device.

In HART the EDD application should take care to read with command 74 the list of the existing modules.

Lexical structure

SCAN method_reference

The attribute of SCAN is specified in Table 72.

Table 72 – SCAN attribute

Usage	Attribute	Description
s	method_reference	Reference to the scan method.

7.8.2.10 SCAN_LIST

Purpose

The SCAN_LIST attribute is a reference to the result list of sub-components that is created by the scan method.

In HART the EDD application should take care to read with command 74 the list of the existing modules.

Lexical structure

SCAN_LIST reference

The attribute of SCAN_LIST is specified in Table 73.

Table 73 – SCAN_LIST attribute

Usage	Attribute	Description
s	reference	Reference to the scan list

7.8.2.11 BYTE_ORDER

Purpose

The BYTE_ORDER attribute indicates the byte order used when sending communication messages between components.

Lexical structure

BYTE_ORDER (BIG_ENDIAN, LITTLE_ENDIAN)

The attributes of BYTE_ORDER are specified in Table 74.

Table 74 – BYTE_ORDER attributes

Usage	Attribute	Description
s	BIG_ENDIAN	High-order bytes are transmitted before the low-order bytes.
s	LITTLE_ENDIAN	Low-order bytes are transmitted before the high-order bytes.

7.8.2.12 CONNECTION_POINT

Purpose

The CONNECTION_POINT attribute specifies a reference to a COLLECTION that describes attributes of a Connection Point. The COLLECTION shall refer to VARIABLE definitions representing Information Model specified Connection Point attributes.

Lexical structure

CONNECTION_POINT reference

The attribute of CONNECTION_POINT is specified in Table 75.

Table 75 – CONNECTION_POINT attribute

Usage	Attribute	Description
m	collection_reference	Reference to a COLLECTION instance.

7.8.2.13 PRODUCT_URI

Purpose

The PRODUCT_URI is a string attribute. The string is a reference to a Communication Server product URI that enables the Server to identify the Communication Server based on the OPC UA Discovery service (see IEC 62541-4). The attribute value corresponds to RegisterServer argument RegisteredServer:serverUri. The product URI shall contain the company name and the product name.

Lexical structure

PRODUCT_URI string

The attribute PRODUCT_URI is specified in Table 76.

Table 76 – PRODUCT_URI attribute

Usage	Attribute	Description
m	string	String literal used as reference to a Communication Server product URI.

7.9 COMPONENT_FOLDER

Purpose

A COMPONENT_FOLDER describes a folder in a catalog. Folders can include other folders. Folders allow to group components for example components of a product family. The LABELs of the folders are associated with the COMPONENT_PATH.

Lexical structure

COMPONENT_FOLDER Identifier [LABEL, CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, HELP, PROTOCOL]

The attributes of COMPONENT_FOLDER are specified in Table 77.

Table 77 – COMPONENT_FOLDER attributes

Usage	Attribute	Description
m	LABEL	Lexical element (see 7.36.9).
o	CLASSIFICATION	Lexical element (see 7.36.1).
o	COMPONENT_PARENT	Lexical element (see 7.36.2).
o	COMPONENT_PATH	Lexical element (see 7.36.3).
o	HELP	Lexical element (see 7.36.8).
o	PROTOCOL	Lexical element (see 7.36.13).

7.10 COMPONENT_REFERENCE

Purpose

A COMPONENT_REFERENCE refers to another component via the COMPONENT_PARENT attribute or with the PROTOCOL, CLASSIFICATION, MANUFACTURER and COMPONENT_PATH attributes. If a folder is referenced, all components that are included in this folder are referenced.

If the COMPONENT_REFERENCE resolves to a folder containing a single EDD, then the COMPONENT_REFERENCE is for a single device. Otherwise a family of devices is indicated.

Alternatively to CLASSIFICATION it is possible to reference a component with DEVICE_TYPE and DEVICE_REVISION.

Lexical structure

COMPONENT_REFERENCE identifier [CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, DEVICE_REVISION, DEVICE_TYPE, MANUFACTURER, PROTOCOL]

The attributes of COMPONENT_FOLDER are specified in Table 78.

Table 78 – COMPONENT_REFERENCE attributes

Usage	Attribute	Description
o	CLASSIFICATION	Lexical element (see 7.36.1).
o	COMPONENT_PARENT	Lexical element (see 7.36.2).
o	COMPONENT_PATH	Lexical element (see 7.36.3).
o	DEVICE_REVISION	Lexical element (see 7.2.2.2).
o	DEVICE_TYPE	Lexical element (see 7.2.2.3).
o	MANUFACTURER	Lexical element (see 7.2.2.6), defaults to MANUFACTURER of EDD.
o	PROTOCOL	Lexical element (see 7.36.13).

7.11 COMPONENT_RELATION

7.11.1 General structure

Purpose

A COMPONENT_RELATION specifies a relation between a module and one or more other modules and the constraints that shall be met to instantiate those other modules.

Lexical structure

COMPONENT_RELATION identifier [COMPONENTS, LABEL, RELATION_TYPE, ADDRESSING, HELP, MAXIMUM_NUMBER, MINIMUM_NUMBER, REQUIRED_INTERFACE, SUPPLIED_INTERFACE]

The attributes of COMPONENT_RELATION are specified in Table 79.

Table 79 – COMPONENT_RELATION attributes

Usage	Attribute	Description
m	COMPONENTS	Lexical element (see 7.11.2.1).
m	LABEL	Lexical element (see 7.36.9).
m	RELATION_TYPE	Lexical element (see 7.11.2.2).
o	ADDRESSING	Lexical element (see 7.11.2.3).
o	HELP	Lexical element (see 7.36.8).
o	MAXIMUM_NUMBER	Lexical element (see 7.11.2.4).
o	MINIMUM_NUMBER	Lexical element (see 7.11.2.5).
o	REQUIRED_INTERFACE	Lexical element (see 7.11.2.6).
o	SUPPLIED_INTERFACE	Lexical element (see 7.36.15).

7.11.2 Specific attributes

7.11.2.1 COMPONENTS

Purpose

The COMPONENTS attribute lists the allowed modules or modular devices and the constraints for their use.

Lexical structure

COMPONENTS [component-reference, AUTO_CREATE, MINIMUM_NUMBER, MAXIMUM_NUMBER, FILTER, REQUIRED_RANGES]<exp>+

The attributes of COMPONENTS are specified in Table 80.

Table 80 – COMPONENTS attributes

Usage	Attribute	Description
m	component-reference	COMPONENT_FOLDER or COMPONENT in the relationship.
o	AUTO_CREATE	Number of automatically created components. Lexical structure AUTO_CREATE (expression)<exp>
o	FILTER	Used if the reference refers to a COMPONENT_FOLDER and specifies a boolean expression that filters the components. Components are included if the expression is TRUE. To filter special COMPONENTs, references to component type are used in the expression. The condition is evaluated for each type contained in the COMPONENT_FOLDER. Lexical structure FILTER (TRUE, FALSE)<exp>
o	MAXIMUM_NUMBER	Maximum allowed number of this component. Lexical structure MAXIMUM_NUMBER (expression)<exp>
o	MINIMUM_NUMBER	Minimum required number of this component. Lexical structure MINIMUM_NUMBER (expression)<exp>
o	REQUIRED_RANGES	Variables with ranges. In a valid configuration the variables have to be in the specified range. Typically this is used to define restrictions for the addressing (e.g. slot). Lexical structure REQUIRED_RANGES [reference, [MAX_VALUE, m]*, [MIN_VALUE, n]*]+

7.11.2.2 RELATION_TYPE

Purpose

The RELATION_TYPE attribute defines the relation between this component and others (child, parent or sibling).

Lexical structure

RELATION_TYPE [CHILD_COMPONENT, PARENT_COMPONENT, SIBLING_COMPONENT]

The attributes of RELATION_TYPE are specified in Table 81.

Table 81 – RELATION_TYPE attributes

Usage	Attribute	Description
s	CHILD_COMPONENT	List of components that are allowed sub-modules.
s	PARENT_COMPONENT	List of components that are allowed parent modules or modular devices for this component.
s	SIBLING_COMPONENT	Relation between siblings that can require or exclude other components. Requiring is possible with MINIMUM_NUMBER > 0. Exclusion is possible with MAXIMUM_NUMBER = 0.
s	NEXT_COMPONENT	Special sibling relation to the next component related to the ADDRESSING. NEXT_COMPONENT is only allowed to use if the ADDRESSING is specified. Not applicable for HART.
s	PREV_COMPONENT	Special sibling relation to the previous component related to the ADDRESSING. PREV_COMPONENT is only allowed to use if the ADDRESSING is specified. Not applicable for HART.

7.11.2.3 ADDRESSING**Purpose**

The ADDRESSING attribute specifies a list of variables that addresses a component uniquely.

Lexical structure

ADDRESSING variable-reference+

The attribute of ADDRESSING is specified in Table 82.

Table 82 – ADDRESSING attribute

Usage	Attribute	Description
m	variable-reference	Reference to a variable.

7.11.2.4 MAXIMUM_NUMBER**Purpose**

The MAXIMUM_NUMBER attribute specifies the number of components that are allowed. The MAXIMUM_NUMBER defaults to 1.

Lexical structure

MAXIMUM_NUMBER (integer)<exp>

The attribute of MAXIMUM_NUMBER is specified in Table 83.

Table 83 – MAXIMUM_NUMBER attribute

Usage	Attribute	Description
o	integer	Total maximum allowed number of all listed components.

7.11.2.5 MINIMUM_NUMBER

Purpose

The MINIMUM_NUMBER attribute specifies the number of components that are required. The MINIMUM_NUMBER defaults to 0.

Lexical structure

MINIMUM_NUMBER (integer)<exp>

The attribute of MINIMUM_NUMBER is specified in Table 84.

Table 84 – MINIMUM_NUMBER attribute

Usage	Attribute	Description
o	integer	Total minimum required number of all listed components.

7.11.2.6 REQUIRED_INTERFACE

Purpose

The REQUIRED_INTERFACE attribute lists the required interfaces of the associated component.

Lexical structure

REQUIRED_INTERFACE interface-reference

The attribute of REQUIRED_INTERFACE is specified in Table 85.

Table 85 – REQUIRED_INTERFACE attribute

Usage	Attribute	Description
m	interface-reference	Reference to an interface.

7.12 CONNECTION (void)

7.13 DOMAIN (void)

7.14 EDIT_DISPLAY

7.14.1 General structure

Purpose

An EDIT_DISPLAY defines how data will be managed for display and editing purposes. It is used to group items together for editing.

This construct is included for backwards compatibility. For future implementation the more general basic construct MENU is recommended.

Lexical structure

EDIT_DISPLAY identifier [EDIT_ITEMS, LABEL, DISPLAY_ITEMS, PRE_EDIT_ACTIONS, POST_EDIT_ACTIONS]

The attributes of EDIT_DISPLAY are specified in Table 86.

Table 86 – EDIT_DISPLAY attributes

Usage	Attribute	Description
m	EDIT_ITEMS	Lexical element (see 7.14.2.1).
m	LABEL	Lexical element (see 7.36.9).
o	DISPLAY_ITEMS	Lexical element (see 7.14.2.2).
o	POST_EDIT_ACTIONS	Lexical element (see 7.14.2.3).
o	PRE_EDIT_ACTIONS	Lexical element (see 7.14.2.4).

7.14.2 Specific attributes

7.14.2.1 EDIT_ITEMS

Purpose

The EDIT_ITEMS attribute defines the set of data items which shall be presented to the user, and may be edited by the user: VARIABLES, WRITE_AS_ONE relations, BLOCK_A PARAMETERS, and elements of BLOCK_A PARAMETERS (i.e. fields or elements of RECORD or VALUE_ARRAY type BLOCK_A PARAMETERS, respectively).

NOTE Specific details for the presentation and editing of a data item (for example, default value, scaling factor, range) are defined within the definitions of each referenced individual item.

If a WRITE_AS_ONE relation appears as an EDIT_ITEM, the user should examine all the variables in the WRITE_AS_ONE relation. The user need not modify all the values but should at least validate that the current values are acceptable.

Lexical structure

EDIT_ITEMS [(reference) <exp>] +

The attribute of EDIT_ITEMS is specified in Table 87.

Table 87 – EDIT_ITEMS attribute

Usage	Attribute	Description
m	reference	Reference to a VARIABLE, a WRITE_AS_ONE, a BLOCK_A PARAMETERS instance or an element of BLOCK_A PARAMETERS. The allowed references are profile specific.

7.14.2.2 DISPLAY_ITEMS

Purpose

The DISPLAY_ITEMS attribute defines the set of data items which shall be presented to the user for informational purposes, i.e., they shall not be edited: VARIABLES, BLOCK_A PARAMETERS, and elements of BLOCK_A PARAMETERS (i.e. fields or elements of RECORD or VALUE_ARRAY type BLOCK_A PARAMETERS, respectively).

NOTE Specific details for the presentation of a data item (for example, default value, scaling factor, range) are defined within the definitions of each referenced individual item.

Lexical structure

DISPLAY_ITEMS [(reference) <exp>] +

The attribute of DISPLAY_ITEM is specified in Table 88.

Table 88 – DISPLAY_ITEM attribute

Usage	Attribute	Description
m	reference	Reference to a VARIABLE, a WRITE_AS_ONE, a BLOCK_A PARAMETERS instance or an element of BLOCK_A PARAMETERS. The allowed references are profile specific.

7.14.2.3 POST_EDIT_ACTIONS

Purpose

The POST_EDIT_ACTIONS attribute specifies METHODS that shall be executed after the user has finished processing the EDIT_DISPLAY. If the EDIT_DISPLAY is aborted, POST_EDIT_ACTIONS are not executed. The specified METHODS shall be executed in the order they appear. If a METHOD exits abnormally, the following METHODS are not executed.

Any POST_EDIT_ACTIONS of a referenced entity in the EDIT_ITEMS shall be executed after a new value is entered. The POST_EDIT_ACTIONS of the EDIT_DISPLAY are executed only after all these individual POST_EDIT_ACTIONS have been completed.

The POST_EDIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

POST_EDIT_ACTIONS [reference]+, DEFINITION

The attributes of POST_EDIT_ACTIONS are specified in Table 89.

7.14.2.4 PRE_EDIT_ACTIONS

Purpose

The PRE_EDIT_ACTIONS attribute specifies METHODS that shall be executed immediately when the EDIT_DISPLAY is activated. The specified METHODS shall be executed in the order they appear. If a METHOD exits abnormally, the following METHODS are not executed and EDIT_DISPLAY activation is cancelled.

The PRE_EDIT_ACTIONS of the EDIT_DISPLAY shall be executed before any of the defined PRE_EDIT_ACTIONS of the referenced entities.

The PRE_EDIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

NOTE PRE_EDIT_ACTIONS of all associated VARIABLE objects are executed immediately before the user actually starts to edit them, for example by setting a cursor in an edit field. Displaying VARIABLE objects without editing does not fire the PRE_EDIT_ACTIONS of the VARIABLE objects.

Lexical structure

PRE_EDIT_ACTIONS [reference]+, DEFINITION

The attributes of PRE_EDIT_ACTIONS are specified in Table 89.

Table 89 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS attributes

Usage	Attribute	Description
m	reference	Reference to a METHOD instance or a METHOD instance with arguments.
o	DEFINITION	Lexical element (see 7.36.4).

7.15 FILE

7.15.1 General structure

Purpose

FILE describes an area of persistent storage that is maintained by the EDD application. The store shall only contain data. The MEMBERS attribute defines the structure of the store, which shall be fixed, that means the MEMBERS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT.

Lexical structure

FILE identifier [MEMBERS, HANDLING, HELP, IDENTITY, LABEL, SHARED, ON_UPDATE_ACTIONS]

The attributes of FILE are specified in Table 90.

Table 90 – FILE attributes

Usage	Attribute	Description
m	MEMBERS	Lexical element (see 7.36.12).
o	HANDLING	Lexical element (see 7.36.6).
o	HELP	Lexical element (see 7.36.8).
o	IDENTITY	Lexical element (see 7.36.21). Specifies the name of the associated file, for example a repository of precalculated engineering data (e.g., tank geometries or flow tables).
o	LABEL	Lexical element (see 7.36.9).
o	SHARED	Lexical element (see 7.15.2.1).
o	ON_UPDATE_ACTIONS	Lexical element (see 7.15.2.2).

7.15.2 Specific attributes

7.15.2.1 SHARED

Purpose

The SHARED attribute specifies whether the file is shared across all instances of the device type. If SHARED is not specified or FALSE the file is associated with the specific device instance.

Lexical structure

SHARED (TRUE, FALSE)

The attributes of SHARED are specified in Table 91.

Table 91 – SHARED attributes

Usage	Attribute	Description
s	TRUE	File is shared within all instances of the same device type.
s	FALSE	File is device instance specific. The default value of SHARED is FALSE.

7.15.2.2 ON_UPDATE_ACTIONS

Purpose

ON_UPDATE_ACTIONS shall only be supported by device simulators.

The ON_UPDATE_ACTIONS attribute specifies METHODS that shall be executed if the shared file is replaced or updated by an external, foreign application. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

Lexical structure

ON_UPDATE_ACTIONS [(reference)<exp>]+

The attribute of ON_UPDATE_ACTIONS is specified in Table 92.

Table 92 – ON_UPDATE_ACTIONS attribute

Usage	Attribute	Description
m	reference	Reference to a METHOD.

7.16 GRAPH

7.16.1 General structure

Purpose

GRAPH describes a graph used to display data from a device in the EDD application. A GRAPH is used to display a data set that is stored in a device, as opposed to a CHART that is used to display continuously collected data values from a device.

Lexical structure

GRAPH identifier [MEMBERS, CYCLE_TIME, HEIGHT, HELP, LABEL, VALIDITY, VISIBILITY, WIDTH, X_AXIS]

The attributes of GRAPH are specified in Table 93.

Table 93 – GRAPH attributes

Usage	Attribute	Description
m	MEMBERS	Lexical element (see 7.36.12).
o	CYCLE_TIME	Lexical element (see 7.16.2.1).
o	HEIGHT	Lexical element (see 7.36.7).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	WIDTH	Lexical element (see 7.36.17).
o	X_AXIS	Lexical element (see 7.16.2.2).

7.16.2 Specific attributes

7.16.2.1 CYCLE_TIME

Purpose

The CYCLE_TIME attribute specifies the length of time, in ms, the EDD application shall wait between successive reads of the source values in the device. By default, a GRAPH has no cycle time.

Lexical structure

CYCLE_TIME (integer) <exp>

The attribute of CYCLE_TIME is specified in Table 94.

Table 94 – CYCLE_TIME attribute

Usage	Attribute	Description
m	integer	Length of time, in ms, between successive reads of the source values in the device. After the GRAPH is displayed by an EDD application, the CYCLE_TIME shall not change.

7.16.2.2 X_AXIS

Purpose

The X_AXIS attribute specifies the way in which the X_AXIS is displayed. Each WAVEFORM on a GRAPH shares a common X_AXIS but each WAVEFORM on a GRAPH has its own Y_AXIS. By default, the display of the X-axis of a GRAPH without an X_AXIS attribute is inferred from the WAVEFORMs on the GRAPH.

Lexical structure

X_AXIS (reference) <exp>

The attribute of X_AXIS is specified in Table 95.

Table 95 – X_AXIS attribute

Usage	Attribute	Description
m	reference	Reference to an AXIS instance.

7.17 GRID

7.17.1 General structure

Purpose

GRID describes a matrix of data from a device to be displayed by the EDD application. A GRID is used to display vectors of data along with the heading or description of the data in that vector. The vectors are displayed horizontally (rows) or vertically (columns) as specified by the ORIENTATION attribute.

Lexical structure

GRID identifier [VECTORS, HANDLING, HEIGHT, HELP, LABEL, ORIENTATION, VALIDITY, VISIBILITY, WIDTH]

The attributes of GRID are specified in Table 96.

Table 96 – GRID attributes

Usage	Attribute	Description
m	VECTORS	Lexical element (see 7.17.2.1).
o	HANDLING	Lexical element (see 7.36.6).
o	HEIGHT	Lexical element (see 7.36.7).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	ORIENTATION	Lexical element (see 7.17.2.2).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	WIDTH	Lexical element (see 7.36.17).

7.17.2 Specific attributes

7.17.2.1 VECTORS

Purpose

The VECTORS attribute specifies the contents of the GRID. Each VECTOR is displayed in order. For a vertical GRID, the vectors are displayed left to right. For a horizontal GRID, the vectors are displayed top to bottom. Each vector consists of a description that provides a label for the vector followed by the data to be displayed.

Lexical structure

VECTORS [description, (reference, string, double, float, integer, unsigned_integer)<exp>+]

The attributes of VECTORS are specified in Table 97.

Table 97 – VECTORS attributes

Usage	Attribute	Description
m	description	String containing a short description of the data within the vector.
o	reference	Reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance, an element of a REFERENCE_ARRAY instance, a VALUE_ARRAY instance or a REFERENCE_ARRAY instance.
	string	String constant or a reference that resolves to a string.
o	double	Double precision floating-point constant.
o	float	Single precision floating-point constant.
o	integer	Integer constant.
o	unsigned_integer	Unsigned integer constant.

7.17.2.2 ORIENTATION**Purpose**

The ORIENTATION attribute specifies the direction in which the vectors are displayed. By default, a GRID without an ORIENTATION attribute is displayed vertically.

Lexical structure

ORIENTATION (HORIZONTAL, VERTICAL) <exp>

The attributes of ORIENTATION are specified in Table 98.

Table 98 – ORIENTATION attributes

Usage	Attribute	Description
s	HORIZONTAL	Vectors are displayed in rows.
s	VERTICAL	Vectors are displayed in columns.

7.18 IMAGE**7.18.1 General structure****Purpose**

IMAGE describes a visual image.

Lexical structure

IMAGE identifier [PATH, HELP, LABEL, LINK, VALIDITY, VISIBILITY]

The attributes of IMAGE are specified in Table 99.

Table 99 – IMAGE attributes

Usage	Attribute	Description
m	PATH	Lexical element (see 7.18.2.1).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	LINK	Lexical element (see 7.18.2.2).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).

7.18.2 Specific attributes

7.18.2.1 PATH

Purpose

The PATH attribute specifies the filename of the image. Images for multiple locales may be specified using the same localization technique used with string literals. The zz country code may be used to specify a low resolution version of the image, which is useful for EDD applications that run on mobile devices having a limited memory.

EXAMPLE "en|english.jpeg|de|german.jpeg|en zz|english-lowres.jpeg|de zz|german-lowres.jpeg"

Lexical structure

PATH (file-name)<exp>

The attribute of PATH is specified in Table 100.

Table 100 – PATH attribute

Usage	Attribute	Description
m	file-name	String literal that identifies the filename of the image.

7.18.2.2 LINK

Purpose

The LINK attribute specifies a MENU or a METHOD that can be accessed from the IMAGE. This allows additional information to be provided about the IMAGE.

Lexical structure

LINK (reference)<exp>

The attribute of LINK is specified in Table 101.

Table 101 – LINK attribute

Usage	Attribute	Description
m	reference	Reference to either a MENU instance or a METHOD instance.

7.19 IMPORT

7.19.1 General structure

Purpose

The EDDL constructs are sufficient for describing any single device. However, additional mechanisms are required to describe multiple revisions of a single device or to describe parts of a generic device. To provide these mechanisms, one EDD can import another EDD. The imported EDD itself can also import other EDDs.

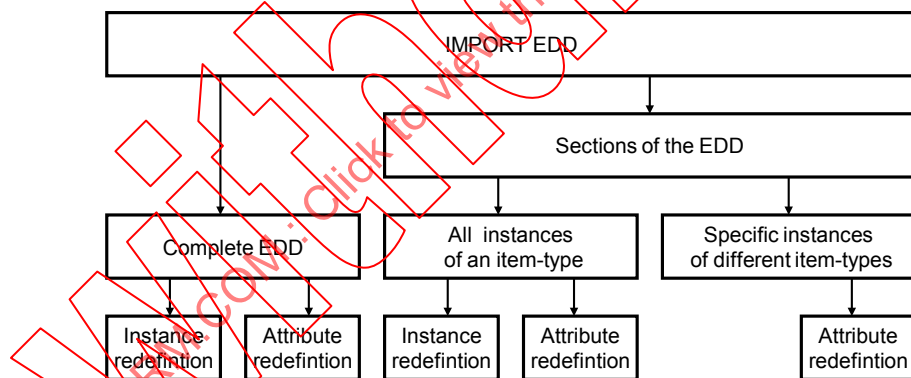
With these mechanisms the EDD of a new device revision can be specified by simply importing the EDD of the old device revision and specifying the changes of the items.

NOTE This type of EDD is called delta description, because the entire EDD is specified as changes to an existing EDD.

Three import types shall be supported:

- importing all items, where all items of the external EDD are imported;
- importing items of a specified type, where only the items of the specified types are imported. If a specified type is imported, further imports of specific items of the same type are not allowed;
- importing a specific item.

The identifier of an imported EDD element shall not be changed. Figure 29 shows the EDDL import mechanisms.



IEC

Figure 29 – EDDL import mechanisms

An alternative way to import is to reference the EDD by file name.

Lexical structure

```

IMPORT ((DD_REVISION, DEVICE_REVISION, DEVICE_TYPE, EDD_PROFILE,
EDD_VERSION, MANUFACTURER, MANUFACTURER_EXT), FILE_NAME), (EVERYTHING,
(AXES, BLOBS, BLOCKS_A, BLOCKS_B, CHART, COLLECTIONS, COMMANDS, COMPONENTS,
COMPONENT_FOLDERS, COMPONENT_REFERENCES, COMPONENT_RELATIONS,
EDIT_DISPLAYS, FILES, GRAPHS, GRIDS, IMAGES, INTERFACES, LISTS, MENUS,
METHODS, PLUGINS, RECORDS, REFERENCE_ARRAYS, REFRESHES, RELATIONS,
RESPONSE_CODES, SOURCES, TEMPLATES, UNITS, VALUE_ARRAYS, VARIABLES,
VARIABLE_LISTS, WAVEFORMS, WRITE_AS_ONES, reference*,
[DELETE identifier]*,
[REDEFINE identifier]*,
attribute-redefinition*))
  
```

The attributes of IMPORT are specified in Table 102.

Table 102 – Importing Device Description

Usage	Attribute	Description
o	attribute-redefinition	See 7.19.2.
o	AXES	All AXIS instances of an external EDD are imported.
o	BLOBS	All BLOB instances of an external EDD are imported.
o	BLOCKS_A	All BLOCK_A instances of an external EDD are imported.
o	BLOCKS_B	All BLOCK_B instances of an external EDD are imported.
o	CHARTS	All CHART instances of an external EDD are imported.
o	COLLECTIONS	All COLLECTION instances of an external EDD are imported.
o	COMMANDS	All COMMAND instances of an external EDD are imported.
o	COMPONENTS	All COMPONENT instances of an external EDD are imported.
o	COMPONENT_FOLDER S	All COMPONENT_FOLDER instances of an external EDD are imported.
o	COMPONENT_REFEREN CES	All COMPONENT_REFERENCE instances of an external EDD are imported.
o	COMPONENT_RELATIO NS	All COMPONENT_RELATION instances of an external EDD are imported.
m	DD_REVISION	See 7.2.2.1.
o	DELETE	An imported instance shall be deleted.
m	DEVICE_REVISION	See 7.2.2.2.
m	DEVICE_TYPE	See 7.2.2.3.
o	EDD_PROFILE	See 7.2.2.4.
o	EDD_VERSION	See 7.2.2.5.
o	EDIT_DISPLAYS	All EDIT_DISPLAY instances of an external EDD are imported.
o	EVERYTHING	All items of an external EDD are imported. If this form is used, no item types and no single instances can be selected. The attributes DELETE, REDEFINE and attribute-redefinition may be used.
o	FILE_NAME	A string which contains a file name and its path in a directory structure. The file contains the EDD which shall be imported. Using this kind of referencing, the specification of the EDD identification information is not necessary.
o	FILES	All FILE instances of an external EDD are imported.
o	GRAPHS	All GRAPH instances of an external EDD are imported.
o	GRIDS	All GRID instances of an external EDD are imported.
o	IMAGES	All IMAGE instances of an external EDD are imported.
o	INTERFACES	All INTERFACE instances of an external EDD are imported.
o	LISTS	All LIST instances of an external EDD are imported.
m	MANUFACTURER	See 7.2.2.6.
c	MANUFACTURER_EXT	See 7.2.2.7.
o	MENUS	All MENU instances of an external EDD are imported.
o	METHODS	All METHOD instances of an external EDD are imported.
o	PLUGINS	All PLUGIN instances of an external EDD are imported.
o	RECORDS	All RECORD instances of an external EDD are imported.
o	REDEFINE	All attributes of an imported instance shall be redefined.
o	reference	A single instance is imported. A single instance shall not be imported, if it is already included in an item type import.
o	REFERENCE_ARRAYS	All REFERENCE_ARRAY instances of an external EDD are imported.
o	REFRESHES	All REFRESH instances of an external EDD are imported.
o	RELATIONS	All REFRESH, UNIT and WRITE_AS_ONE instances of an external EDD are imported.
o	RESPONSE_CODES	All RESPONSE_CODES instances of an external EDD are imported.
o	SOURCES	All SOURCE instances of an external EDD are imported.
o	TEMPLATES	All TEMPLATE instances of an external EDD are imported.
o	UNITS	All UNIT instances of an external EDD are imported.

Usage	Attribute	Description
o	VALUE_ARRAYS	All VALUE_ARRAY instances of an external EDD are imported.
o	VARIABLE_LISTS	All VARIABLE_LIST instances of an external EDD are imported.
o	VARIABLES	All VARIABLE instances of an external EDD are imported.
o	WAVEFORMS	All WAVEFORM instances of an external EDD are imported.
o	WRITE_AS_ONES	All WRITE_AS_ONE instances of an external EDD are imported.

7.19.2 Redefinitions

7.19.2.1 General

Purpose

Attributes of an imported instance may be added, deleted or redefined. Adding, deleting and redefining do not affect all attributes of an EDD basic construct element.

The redefinition attributes are specified in Table 103.

Table 103 – Redefinition attributes

Usage	Attribute	Description
s	ADD	Adds the following specified attribute. "A" indicates that ADD can be applied to the attribute in the redefinition rules tables (see Table 104 to Table 133).
s	DELETE	Deletes the following specified attribute. "D" indicates that DELETE can be applied to the attribute in the redefinition rules tables (see Table 104 to Table 133).
s	REDEFINE	Redefines the following specified attribute. "R" indicates that REDEFINE can be applied to the attribute in the redefinition rules tables (see Table 104 to Table 133).

7.19.2.2 AXIS

Lexical structure

AXIS identifier, [[DELETE, REDEFINE], [CONSTANT_UNIT, HELP, LABEL, MAX_VALUE, MIN_VALUE, SCALING]]+

The redefinition rules of AXIS are specified in Table 103 and Table 104.

Table 104 – Redefinition rules for AXIS attributes

A	D	R	Attribute	Description
	•	•	CONSTANT_UNIT	Lexical element (see 7.3.2.1).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	MAX_VALUE	Lexical element (see 7.3.2.2).
	•	•	MIN_VALUE	Lexical element (see 7.3.2.3).
	•	•	SCALING	Lexical element (see 7.3.2.4).

7.19.2.3 BLOB

Lexical structure

BLOB identifier, [[DELETE, REDEFINE], [HANDLING, HELP, IDENTITY, LABEL]]+

The redefinition rules of BLOB are specified in Table 103 and Table 105.

Table 105 – Redefinition rules for BLOB attributes

A	D	R	Attribute	Description
		•	HANDLING	Lexical element (see 7.36.6).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	IDENTITY	Lexical element (see 7.36.21).
	•	•	LABEL	Lexical element (see 7.36.9).

7.19.2.4 BLOCK_A

Lexical structure

BLOCK identifier, [[ADD, DELETE, REDEFINE], [CHARACTERISTICS, LABEL, PARAMETERS, parameters-list-element, AXIS_ITEMS, CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISPLAY_ITEMS, FILE_ITEMS, GRAPH_ITEMS, GRID_ITEMS, HELP, IMAGE_ITEMS, LIST_ITEMS, LOCAL_PARAMETERS, local-parameters-list-element, MENU_ITEMS, METHOD_ITEMS, PARAMETER_LISTS, parameter-lists-list-element, PLUGIN_ITEMS, REFERENCE_ARRAY_ITEMS, REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS, WRITE_AS_ONE_ITEMS, CHARTS, charts-element, FILES, files-element, LISTS, lists-element, GRAPHS, graphs-element, GRIDS, grids-element, MENUS, menus-element, METHODS, methods-element, PLUGINS, plugins-element]]+

The redefinition rules of BLOCK_A are specified in Table 103 and Table 106.

Table 106 – Redefinition rules for BLOCK_A attributes

A	D	R	Attribute	Description
		•	CHARACTERISTICS	Lexical element (see 7.4.1.2.1).
		•	LABEL	Lexical element (see 7.36.9).
		•	PARAMETERS	Lexical element (see 7.4.1.2.2).
•		•	parameters-list-element	Single list elements of PARAMETERS may be changed.
	•	•	AXIS_ITEMS	Lexical element (see 7.4.1.2.3).
	•	•	CHART_ITEMS	Lexical element (see 7.4.1.2.4).
	•	•	COLLECTION_ITEMS	Lexical element (see 7.4.1.2.5).
	•	•	EDIT_DISPLAY_ITEMS	Lexical element (see 7.4.1.2.6).
	•	•	FILE_ITEMS	Lexical element (see 7.4.1.2.7).
	•	•	GRAPH_ITEMS	Lexical element (see 7.4.1.2.8).
	•	•	GRID_ITEMS	Lexical elements (see 7.4.1.2.9).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	IMAGE_ITEMS	Lexical elements (see 7.4.1.2.10).
	•	•	LIST_ITEMS	Lexical element (see 7.4.1.2.11).
		•	LOCAL_PARAMETERS	Lexical element (see 7.4.1.2.12).
•		•	local-parameters-list-element	Single list elements of LOCAL_PARAMETERS may be changed.
	•	•	MENU_ITEMS	Lexical element (see 7.4.1.2.13).
	•	•	METHOD_ITEMS	Lexical element (see 7.4.1.2.14).
		•	PARAMETER_LISTS	Lexical element (see 7.4.1.2.15).
•		•	parameter-lists-list-element	Single list elements of PARAMETERS_LISTS may be changed.
	•	•	PLUGIN_ITEMS	Lexical element (see 7.4.1.2.29).
	•	•	REFERENCE_ARRAY_ITEMS	Lexical element (see 7.4.1.2.16).
	•	•	REFRESH_ITEMS	Lexical element (see 7.4.1.2.17).
	•	•	SOURCE_ITEMS	Lexical element (see 7.4.1.2.18).
	•	•	UNIT_ITEMS	Lexical element (see 7.4.1.2.19).
	•	•	WAVEFORM_ITEMS	Lexical element (see 7.4.1.2.20).
	•	•	WRITE_AS_ONE_ITEMS	Lexical element (see 7.4.1.2.21).
	•	•	CHARTS	Lexical element (see 7.4.1.2.22).
•	•	•	charts-element	Single elements of CHARTS may be changed.
	•	•	FILES	Lexical element (see 7.4.1.2.28).
•	•	•	files-element	Single elements of FILES may be changed.
	•	•	LISTS	Lexical element (see 7.4.1.2.23).
•	•	•	lists-element	Single elements of LISTS may be changed.
	•	•	GRAPHS	Lexical element (see 7.4.1.2.24).
•	•	•	graphs-element	Single elements of GRAPHS may be changed.
	•	•	GRIDS	Lexical element (see 7.4.1.2.25).
•	•	•	grids-element	Single elements of GRIDS may be changed.
	•	•	MENUS	Lexical element (see 7.4.1.2.26).
•	•	•	menus-element	Single elements of MENUS may be changed.
	•	•	METHODS	Lexical element (see 7.4.1.2.27).
•	•	•	methods-element	Single elements of METHODS may be changed.
	•	•	PLUGINS	Lexical element (see 7.4.1.2.30).
•	•	•	plugins-element	Single elements of PLUGIN may be changed.

7.19.2.5 BLOCK_B

Lexical structure

BLOCK identifier, [REDEFINE, [NUMBER, TYPE]]+

The redefinition rules of BLOCK_B are specified in Table 103 and Table 107.

Table 107 – Redefinition rules for BLOCK_B attributes

A	D	R	Attribute	Description
		•	NUMBER	Lexical element (see 7.4.2.2.1).
		•	TYPE	Lexical element (see 7.4.2.2.2).

7.19.2.6 CHART

Lexical structure

CHART identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, CYCLE_TIME, HEIGHT, HELP, LABEL, LENGTH, TYPE, VALIDITY, VISIBILITY, WIDTH]]+

The redefinition rules of CHART are specified in Table 103 and Table 108.

Table 108 – Redefinition rules for CHART attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 7.36.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	CYCLE_TIME	Lexical element (see 7.5.2.1).
	•	•	HEIGHT	Lexical element (see 7.36.7).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	LENGTH	Lexical element (see 7.5.2.2).
	•	•	TYPE	Lexical element (see 7.5.2.3).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	WIDTH	Lexical element (see 7.36.17).

7.19.2.7 COLLECTION

Lexical structure

COLLECTION identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]]+

The redefinition rules of COLLECTION are specified in Table 103 and Table 109.

Table 109 – Redefinition rules for COLLECTION attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 7.36.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	PRIVATE	Lexical element (see 7.36.18).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).

7.19.2.8 COMMAND**Lexical structure**

COMMAND identifier, [[ADD, DELETE, REDEFINE], [OPERATION, TRANSACTION, INDEX, BLOCK_B, NUMBER, SLOT, SUB_SLOT, API, HEADER, RESPONSE_CODES, response-codes-list-element]]+

The redefinition rules of COMMAND are specified in Table 103 and Table 110.

Table 110 – Redefinition rules for COMMAND attributes

A	D	R	Attribute	Description
			OPERATION	Lexical element (see 7.7.2.1).
•	•	•	TRANSACTION	Lexical element (see 7.7.2.2).
	•	•	INDEX	Lexical element (see 7.7.2.3).
			BLOCK_B	Lexical element (see 7.7.2.4).
			NUMBER	Lexical element (see 7.7.2.5).
	•	•	SLOT	Lexical element (see 7.7.2.6).
	•	•	SUB_SLOT	Lexical element (see 7.7.2.7).
	•	•	API	Lexical element (see 7.7.2.7).
	•	•	HEADER	Lexical element (see 7.7.2.9).
	•	•	POST_RECEIVE_ACTIONS	Lexical element (see 7.7.2.12).
	•	•	RESPONSE_CODES	Lexical element (see 7.36.14).
•	•	•	response-codes-list-element	Single list elements of RESPONSE_CODES may be changed.

7.19.2.9 COMPONENT**Lexical structure**

COMPONENT identifier, [[DELETE, REDEFINE], [LABEL, BYTE_ORDER, CAN_DELETE, CHECK_CONFIGURATION, CLASSIFICATION, CONNECTION_POINT, HELP, COMPONENT_PARENT, COMPONENT_PATH, COMPONENT_RELATIONS, DECLARATION, DETECT, EDD, INITIAL_VALUES, PRODUCT_URI, PROTOCOL, REDUNDANCY, SCAN, SCAN_LIST, SUPPLIED_INTERFACE]]+

The redefinition rules of COMPONENT are specified in Table 103 and Table 111.

Table 111 – Redefinition rules for COMPONENT attributes

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 7.36.9).
	•	•	BYTE_ORDER	Lexical element (see 7.8.2.11).
	•	•	CAN_DELETE	Lexical element (see 7.8.2.1).
	•	•	CHECK_CONFIGURATION	Lexical element (see 7.8.2.2).
	•	•	CLASSIFICATION	Lexical element (see 7.36.1).
	•	•	COMPONENT_PARENT	Lexical element (see 7.36.2).
	•	•	COMPONENT_PATH	Lexical element (see 7.36.3).
	•	•	COMPONENT_RELATIONS	Lexical element (see 7.8.2.3).
	•	•	CONNECTION_POINT	Lexical element (see 7.8.2.12).
	•	•	DECLARATION	Lexical element (see 7.8.2.4).
	•	•	DETECT	Lexical element (see 7.8.2.5).
	•	•	EDD	Lexical element (see 7.8.2.6).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	INITIAL_VALUES	Lexical element (see 7.8.2.7).
	•	•	PRODUCT_URI	Lexical element (see 7.8.2.13).
	•	•	PROTOCOL	Lexical element (see 7.36.13).
	•	•	REDUNDANCY	Lexical element (see 7.8.2.8).
	•	•	SCAN	Lexical element (see 7.8.2.9).
	•	•	SCAN_LIST	Lexical element (see 7.8.2.10).
	•	•	SUPPLIED_INTERFACE	Lexical element (see 7.36.15).

7.19.2.10 COMPONENT_FOLDER

Lexical structure

COMPONENT_FOLDER identifier, [[DELETE, REDEFINE], [LABEL, CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, HELP, PROTOCOL]]+

The redefinition rules of COMPONENT_FOLDER are specified in Table 103 and Table 112.

Table 112 – Redefinition rules for COMPONENT_FOLDER attributes

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 7.36.9).
	•	•	CLASSIFICATION	Lexical element (see 7.36.1).
	•	•	COMPONENT_PARENT	Lexical element (see 7.36.2).
	•	•	COMPONENT_PATH	Lexical element (see 7.36.3).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	PROTOCOL	Lexical element (see 7.36.13).

7.19.2.11 COMPONENT_REFERENCE

Lexical structure

COMPONENT_REFERENCE identifier, [[DELETE, REDEFINE], [CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, DEVICE_REVISION, DEVICE_TYPE, MANUFACTURER, PROTOCOL]]+

The redefinition rules of COMPONENT_REFERENCE are specified in Table 103 and Table 113.

Table 113 – Redefinition rules for COMPONENT_REFERENCE attributes

A	D	R	Attribute	Description
	•	•	CLASSIFICATION	Lexical element (see 7.36.1).
	•	•	COMPONENT_PARENT	Lexical element (see 7.36.2).
	•	•	COMPONENT_PATH	Lexical element (see 7.36.3).
	•	•	DEVICE_REVISION	Lexical element (see 7.2.2.2).
	•	•	DEVICE_TYPE	Lexical element (see 7.2.2.3).
	•	•	MANUFACTURER	Lexical element (see 7.2.2.6).
	•	•	PROTOCOL	Lexical element (see 7.36.13).

7.19.2.12 COMPONENT_RELATION

Lexical structure

COMPONENT_RELATION identifier, [[ADD, DELETE, REDEFINE], [COMPONENTS, AUTO_CREATE, FILTER, REQUIRED_RANGES, LABEL, RELATION_TYPE, ADDRESSING, HELP, MAXIMUM_NUMBER, MINIMUM_NUMBER, REQUIRED_INTERFACE, SUPPLIED_INTERFACE]]+

The redefinition rules of COMPONENT_RELATION are specified in Table 103 and Table 114.

Table 114 – Redefinition rules for COMPONENT_RELATION attributes

A	D	R	Attribute	Description
		•	COMPONENTS	Lexical element (see 7.11.2.1).
•	•	•	components-list-element	Single list elements of COMPONENTS may be changed.
	•	•	AUTO_CREATE	Lexical element (see 7.11.2.1).
	•	•	FILTER	Lexical element (see 7.11.2.1).
	•	•	REQUIRED_RANGES	Lexical element (see 7.11.2.1).
		•	LABEL	Lexical element (see 7.36.9).
		•	RELATION_TYPE	Lexical element (see 7.11.2.2).
	•	•	ADDRESSING	Lexical element (see 7.11.2.3).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	MAXIMUM_NUMBER	Lexical element (see 7.11.2.4).
	•	•	MINIMUM_NUMBER	Lexical element (see 7.11.2.5).
	•	•	REQUIRED_INTERFACE	Lexical element (see 7.11.2.6).
	•	•	SUPPLIED_INTERFACE	Lexical element (see 7.36.15).

7.19.2.13 EDIT_DISPLAY

Lexical structure

EDIT_DISPLAY identifier, [[DELETE, REDEFINE], [EDIT_ITEMS, LABEL, DISPLAY_ITEMS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS]]+

The redefinition rules of EDIT_DISPLAY are specified in Table 103 and Table 115.

Table 115 – Redefinition rules for EDIT_DISPLAY attributes

A	D	R	Attribute	Description
		•	EDIT_ITEMS	Lexical element (see 7.14.2.1).
		•	LABEL	Lexical element (see 7.36.9).
	•	•	DISPLAY_ITEMS	Lexical element (see 7.14.2.2).
	•	•	POST_EDIT_ACTIONS	Lexical element (see 7.14.2.3).
	•	•	PRE_EDIT_ACTIONS	Lexical element (see 7.14.2.4).

7.19.2.14 FILE

Lexical structure

FILE identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, HANDLING, HELP, IDENTITY, LABEL, SHARED, ON_UPDATE_ACTIONS]]+

The redefinition rules of FILE are specified in Table 103 and Table 116.

Table 116 – Redefinition rules for FILE attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 7.36.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	HANDLING	Lexical element (see 7.36.6).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	IDENTITY	Lexical element (see 7.36.21).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	SHARED	Lexical element (see 7.15.2.1).
	•	•	ON_UPDATE_ACTIONS	Lexical element (see 7.15.2.2).

7.19.2.15 GRAPH

Lexical structure

GRAPH identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, CYCLE_TIME, HEIGHT, HELP, LABEL, VALIDITY, VISIBILITY, WIDTH, X_AXIS]]+

The redefinition rules of GRAPH are specified in Table 103 and Table 117.

Table 117 – Redefinition rules for GRAPH attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 7.36.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	CYCLE_TIME	Lexical element (see 7.16.2.1).
	•	•	HEIGHT	Lexical element (see 7.36.7).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	VALIDITY	Lexical element (see 7.38.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	WIDTH	Lexical element (see 7.36.17).
	•	•	X_AXIS	Lexical element (see 7.16.2.2).

7.19.2.16 GRID**Lexical structure**

GRID identifier, [[ADD, DELETE, REDEFINE], VECTORS, HANDLING, HEIGHT, HELP, LABEL, ORIENTATION, VALIDITY, VISIBILITY, WIDTH][+]

The redefinition rules of GRAPH are specified in Table 103 and Table 118.

Table 118 – Redefinition rules for GRID attributes

A	D	R	Attribute	Description
		•	VECTORS	Lexical element (see 7.17.2.1).
	•	•	HANDLING	Lexical element (see 7.36.6).
	•	•	HEIGHT	Lexical element (see 7.36.7).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	ORIENTATION	Lexical element (see 7.17.2.2).
	•	•	VALIDITY	Lexical element (see 7.38.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	WIDTH	Lexical element (see 7.36.17).

7.19.2.17 IMAGE**Lexical structure**

IMAGE identifier, [[DELETE, REDEFINE], PATH, HELP, LABEL, LINK, VALIDITY, VISIBILITY][+]

The redefinition rules of GRAPH are specified in Table 103 and Table 119.

Table 119 – Redefinition rules for IMAGE attributes

A	D	R	Attribute	Description
		•	PATH	Lexical element (see 7.18.2.1).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	LINK	Lexical element (see 7.18.2.2).
	•	•	VALIDITY	Lexical element (see 7.38.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).

7.19.2.18 INTERFACE

Lexical structure

INTERFACE identifier, [[DELETE, REDEFINE], [DECLARATION, LABEL, HELP]]+

The redefinition rules of INTERFACE are specified in Table 103 and Table 120.

Table 120 – Redefinition rules for INTERFACE attributes

A	D	R	Attribute	Description
		•	DECLARATION	Lexical element (see 7.20.2).
		•	LABEL	Lexical element (see 7.36.9).
	•	•	HELP	Lexical element (see 7.36.8).

7.19.2.19 LIST

Lexical structure

LIST identifier, [[DELETE, REDEFINE], [TYPE, CAPACITY, COUNT, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]]+

The redefinition rules of LIST are specified in Table 103 and Table 121.

Table 121 – Redefinition rules for LIST attributes

A	D	R	Attribute	Description
		•	TYPE	Lexical element (see 7.22.2.1).
	•	•	CAPACITY	Lexical element (see 7.22.2.2).
	•	•	COUNT	Lexical element (see 7.22.2.3).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	PRIVATE	Lexical element (see 7.36.18).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).

7.19.2.20 MENU

Lexical structure

MENU identifier, [[ADD, DELETE, REDEFINE], [ITEMS, items-list-element, LABEL, ACCESS, HELP, EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS,

POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, STYLE, VALIDITY, VISIBILITY]]+

The redefinition rules of MENU are specified in Table 103 and Table 122.

Table 122 – Redefinition rules for MENU attributes

A	D	R	Attribute	Description
		•	ITEMS	Lexical element (see 7.23.2.1).
		•	LABEL	Lexical element (see 7.36.9).
	•	•	ACCESS	Lexical element (see 7.23.2.2).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	EXIT_ACTIONS	Lexical element (see 7.23.2.12).
	•	•	INIT_ACTIONS	Lexical element (see 7.23.2.13).
	•	•	POST_EDIT_ACTIONS	Lexical element (see 7.23.2.3).
	•	•	POST_READ_ACTIONS	Lexical element (see 7.23.2.4).
	•	•	POST_WRITE_ACTIONS	Lexical element (see 7.23.2.5).
	•	•	PRE_EDIT_ACTIONS	Lexical element (see 7.23.2.6).
	•	•	PRE_READ_ACTIONS	Lexical element (see 7.23.2.7).
	•	•	PRE_WRITE_ACTIONS	Lexical element (see 7.23.2.8).
	•	•	STYLE	Lexical element (see 7.23.2.11).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).

7.19.2.21 METHOD

Lexical structure

METHOD identifier, [[DELETE, REDEFINE], [DEFINITION, ACCESS, CLASS, LABEL, HELP, PRIVATE, VALIDITY, VISIBILITY]]+

The redefinition rules of METHOD are specified in Table 103 and Table 123.

Table 123 – Redefinition rules for METHOD attributes

A	D	R	Attribute	Description
		•	DEFINITION	Lexical element (see 7.36.4).
	•	•	ACCESS	Lexical element (see 7.24.2.1).
		•	CLASS	Lexical element (see 7.24.2.2).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	HELP	Lexical element (see 7.36.8).
			TYPE	Lexical element (see 7.24.2.3). TYPE shall not be redefined.
	•	•	PRIVATE	Lexical element (see 7.36.18).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).

7.19.2.22 PLUGIN

Lexical structure

PLUGIN identifier, [[DELETE, REDEFINE], [LABEL, HELP, UUID, VALIDITY, VISIBILITY]]+

The redefinition rules of PLUGIN are specified in Table 103 and Table 124.

Table 124 – Redefinition rules for PLUGIN attributes

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 7.36.9).
	•	•	HELP	Lexical element (see 7.36.8).
		•	UUID	Lexical element (see 7.42.2).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).

7.19.2.23 RECORD

Lexical structure

RECORD identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, LABEL, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]]+

The redefinition rules of RECORD are specified in Table 103 and Table 125.

Table 125 – Redefinition rules for RECORD attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 7.36.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
		•	LABEL	Lexical element (see 7.36.9).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	PRIVATE	Lexical element (see 7.36.18).
	•	•	RESPONSE_CODES	Lexical element (see 7.36.14).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	WRITE_MODE	Lexical element (see 7.36.20).

7.19.2.24 REFERENCE_ARRAY

Lexical structure

REFERENCE_ARRAY identifier, [[ADD, DELETE, REDEFINE], [ELEMENTS, elements-list-element, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]]+

The redefinition rules of REFERENCE_ARRAY are specified in Table 103 and Table 126.

Table 126 – Redefinition rules for REFERENCE_ARRAY attributes

A	D	R	Attribute	Description
		•	ELEMENTS	Lexical element (see 7.27.2).
•	•	•	elements-list-element	Single list elements of ELEMENTS may be changed.
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	PRIVATE	Lexical element (see 7.36.18).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).

7.19.2.25 RESPONSE_CODES**Lexical structure**

RESPONSE_CODES identifier, [[ADD, DELETE, REDEFINE], response-code-list-element]+

The redefinition rules of RESPONSE_CODES are specified in Table 103 and Table 127.

Table 127 – Redefinition rules for RESPONSE_CODES attributes

A	D	R	Attribute	Description
•	•	•	response-code-list-element	Single list elements of RESPONSE_CODES may be changed.

7.19.2.26 SOURCE**Lexical structure**

SOURCE identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, EMPHASIS, EXIT_ACTIONS, HELP, INIT_ACTIONS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY, VISIBILITY, Y_AXIS]]+

The redefinition rules of SOURCE are specified in Table 103 and Table 128.

Table 128 – Redefinition rules for SOURCE attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 7.36.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	EMPHASIS	Lexical element (see 7.36.5).
	•	•	EXIT_ACTIONS	Lexical element (see 7.30.2.1).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	INIT_ACTIONS	Lexical element (see 7.30.2.2).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	LINE_COLOR	Lexical element (see 7.36.10).
	•	•	LINE_TYPE	Lexical element (see 7.36.11).
	•	•	REFRESH_ACTIONS	Lexical element (see 7.30.2.3).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	Y_AXIS	Lexical element (see 7.30.2.4).

7.19.2.27 TEMPLATE

Lexical structure

TEMPLATE identifier, [[DELETE, REDEFINE], [DEFAULT_VALUES, LABEL, HELP, VALIDITY]]+

The redefinition rules of TEMPLATE are specified in Table 103 and Table 129.

Table 129 – Redefinition rules for TEMPLATE attributes

A	D	R	Attribute	Description
		•	DEFAULT_VALUES	Lexical element (see 7.31.2).
		•	LABEL	Lexical element (see 7.36.9).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	VALIDITY	Lexical element (see 7.36.16).

7.19.2.28 VALUE_ARRAY

Lexical structure

VALUE_ARRAY identifier, [[DELETE, REDEFINE], [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]]+

The redefinition rules of VALUE_ARRAY are specified in Table 103 and Table 130.

Table 130 – Redefinition rules for VALUE_ARRAY attributes

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 7.36.9).
		•	NUMBER_OF_ELEMENTS	Lexical element (see 7.32.2.1).
		•	TYPE	Lexical element (see 7.32.2.2).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	PRIVATE	Lexical element (see 7.36.18).
	•	•	RESPONSE_CODES	Lexical element (see 7.36.14).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	WRITE_MODE	Lexical element (see 7.36.20).

7.19.2.29 VARIABLE

Lexical structure

VARIABLE identifier, [[ADD, DELETE, REDEFINE], [CLASS, LABEL, TYPE, DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, MAX_VALUE, MIN_VALUE, SCALING_FACTOR, TIME_FORMAT, TIME_SCALE, type-list-element, CONSTANT_UNIT, HANDLING, HEIGHT, HELP, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_RQSTUPDATE_ACTIONS, POST_USERCHANGE_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, PRIVATE, REFRESH_ACTIONS, RESPONSE_CODES, VALIDITY, VISIBILITY, WIDTH, WRITE_MODE]]+

The redefinition rules are specified in Table 103 and Table 131.

Table 131 – Redefinition rules for VARIABLE attributes

A	D	R	Attribute	Description
		•	CLASS	Lexical element (see 7.33.2.1).
	•	•	LABEL	Lexical element (see 7.36.9).
			TYPE	Lexical element (see 7.33.2.2). TYPE shall not be redefined.
•	•	•	type-list-element	Single enumeration and bit-enumeration list elements may be changed.
	•	•	DEFAULT_VALUE	Lexical element (see 7.33.2.2).
	•	•	DISPLAY_FORMAT	Lexical element (see 7.33.2.2).
	•	•	EDIT_FORMAT	Lexical element (see 7.33.2.2).
	•	•	INITIAL_VALUE	Lexical element (see 7.33.2.2).
	•	•	MAX_VALUE	Lexical element (see 7.33.2.2).
	•	•	MIN_VALUE	Lexical element (see 7.33.2.2).
	•	•	SCALING_FACTOR	Lexical element (see 7.33.2.2).
	•	•	TIME_FORMAT	Lexical element (see 7.33.2.2).
	•	•	TIME_SCALE	Lexical element (see 7.33.2.2).
	•	•	CONSTANT_UNIT	Lexical element (see 7.33.2.3).
	•	•	HANDLING	Lexical element (see 7.36.6).
	•	•	HEIGHT	Lexical element (see 7.36.7).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	POST_EDIT_ACTIONS	Lexical element (see 7.33.2.6).
	•	•	POST_READ_ACTIONS	Lexical element (see 7.33.2.7).
	•	•	POST_RQSTUPDATE_ACTIONS	Lexical element (see 7.33.2.15).
	•	•	POST_USERCHANGE_ACTIONS	Lexical element (see 7.33.2.16).
	•	•	POST_WRITE_ACTIONS	Lexical element (see 7.33.2.8).
	•	•	PRE_EDIT_ACTIONS	Lexical element (see 7.33.2.9).
	•	•	PRE_READ_ACTIONS	Lexical element (see 7.33.2.10).
	•	•	PRE_WRITE_ACTIONS	Lexical element (see 7.33.2.11).
	•	•	PRIVATE	Lexical element (see 7.36.18).
	•	•	REFRESH_ACTIONS	Lexical element (see 7.33.2.13).
	•	•	RESPONSE_CODES	Lexical element (see 7.36.14).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	WIDTH	Lexical element (see 7.36.17).
	•	•	WRITE_MODE	Lexical element (see 7.36.20).

7.19.2.30 VARIABLE_LIST**Lexical structure**

VARIABLE_LIST identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, HELP, LABEL, RESPONSE_CODES]]+

The redefinition rules of VARIABLE_LIST are specified in Table 103 and Table 132.

Table 132 – Redefinition rules for VARIABLE_LIST attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 7.36.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	RESPONSE_CODES	Lexical element (see 7.36.14).

7.19.2.31 WAVEFORM

Lexical structure

WAVEFORM identifier, [[ADD, DELETE, REDEFINE], [TYPE, X_VALUES, Y_VALUES, NUMBER_OF_POINTS, X_INITIAL, X_INCREMENT, EMPHASIS, HANDLING, HELP, INIT_ACTIONS, KEY_POINTS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY, VISIBILITY, Y_AXIS]]+

The redefinition rules of WAVEFORM are specified in Table 103 and Table 133.

Table 133 – Redefinition rules for WAVEFORM attributes

A	D	R	Attribute	Description
		•	TYPE	Lexical element (see 7.35.2.1).
•		•	X_VALUES	Lexical element (see 7.35.2.1).
•		•	Y_VALUES	Lexical element (see 7.35.2.1).
	•	•	NUMBER_OF_POINTS	Lexical element (see 7.35.2.1).
		•	X_INITIAL	Lexical element (see 7.35.2.1).
		•	X_INCREMENT	Lexical element (see 7.35.2.1).
	•	•	EMPHASIS	Lexical element (see 7.36.5).
	•	•	EXIT_ACTIONS	Lexical element (see 7.35.2.2).
	•	•	HANDLING	Lexical element (see 7.36.6).
	•	•	HELP	Lexical element (see 7.36.8).
	•	•	INIT_ACTIONS	Lexical element (see 7.35.2.3).
	•	•	KEY_POINTS	Lexical element (see 7.35.2.4).
	•	•	LABEL	Lexical element (see 7.36.9).
	•	•	LINE_COLOR	Lexical element (see 7.36.10).
	•	•	LINE_TYPE	Lexical element (see 7.36.11).
	•	•	REFRESH_ACTIONS	Lexical element (see 7.35.2.5).
	•	•	VALIDITY	Lexical element (see 7.36.16).
	•	•	VISIBILITY	Lexical element (see 7.36.19).
	•	•	Y_AXIS	Lexical element (see 7.35.2.6).

7.20 INTERFACE

7.20.1 General structure

Purpose

The INTERFACE specifies a set of basic constructs of components that are accessible by other components. An interface can contain declarations of basic constructs, for example VARIABLES, MENUS.

All supplied INTERFACES of a component should be listed in the SUPPLIED_INTERFACE attribute of the COMPONENT.

If INTERFACES are defined for a COMPONENT, then access is restricted. Other components may only access EDD basic constructs (e.g. VARIABLES) defined in an interface.

If no interface is defined for the component, then access is allowed to all basic constructs.

Lexical structure

INTERFACE identifier [LABEL, HELP, DECLARATION]

The attributes of INTERFACE are specified in Table 134.

Table 134 – INTERFACE attributes

Usage	Attribute	Description
m	DECLARATION	Lexical element (see 7.20.2).
m	LABEL	Lexical element (see 7.36.9).
o	HELP	Lexical element (see 7.36.8).

7.20.2 Specific attribute – DECLARATION

Purpose

The DECLARATION attribute specifies the basic constructs included in the interface.

Lexical structure

DECLARATION [IMAGE, MENU, METHOD, VARIABLE] +

The attributes of DECLARATION are specified in Table 135.

Table 135 – DECLARATION attributes

Usage	Attribute	Description
o	IMAGE	IMAGE declarations (see 7.18).
o	MENU	MENU declarations (see 7.23).
o	METHOD	METHOD declarations (see 7.24).
o	VARIABLE	VARIABLE declarations (see 7.33).

7.21 LIKE

Purpose

LIKE is used to create a new EDD instance based on an existing item. LIKE makes a copy of the existing item type with all attributes.

Lexical structure

LIKE identifier, reference, attribute-redefinition+, item-type

The attributes of LIKE are specified in Table 136.

Table 136 – LIKE attributes

Usage	Attribute	Description
m	identifier	Name of the new EDD instance.
m	reference	Reference to an EDD instance which shall be copied.
o	attribute-redefinition	Specifies whether an attribute is added, deleted or redefined (see 7.19.2). Only existing attributes can be deleted or redefined.
o	item-type	Type of the new EDD instance. If specified, it shall match to the type of the referenced item.

7.22 LIST

7.22.1 General structure

Purpose

LIST describes a sequence of elements where each element of the sequence is defined by an EDDL construct instance.

Lexical structure

LIST identifier [TYPE, CAPACITY, COUNT, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]

The attributes of LIST are specified in Table 137.

Table 137 – LIST attributes

Usage	Attribute	Description
m	TYPE	Lexical element (see 7.22.2.1).
o	CAPACITY	Lexical element (see 7.22.2.2).
o	COUNT	Lexical element (see 7.22.2.3).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	PRIVATE	Lexical element (see 7.36.18).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).

7.22.2 Specific attributes

7.22.2.1 TYPE

Purpose

The TYPE attribute is a reference to an EDDL construct instance (for example, VARIABLE, COLLECTION). All attributes from the EDDL construct apply to each of the LIST elements.

Lexical structure

TYPE reference

The attribute of TYPE is specified in Table 138.

Table 138 – TYPE attribute

Usage	Attribute	Description
m	reference	Reference to a VARIABLE instance, a RECORD instance, a VALUE_ARRAY instance, a COLLECTION instance, a REFERENCE_ARRAY instance or a LIST instance. The item referenced in the TYPE attribute of a LIST shall not contain VALIDITY or any other conditional expressions that would cause data type structures to change. This restriction also applies for contained items. The reference shall terminate to a data item.

7.22.2.2 CAPACITY

Purpose

The CAPACITY attribute specifies the maximum number of elements that may be stored in the list.

Lexical structure

CAPACITY unsigned_integer

The attribute of CAPACITY is specified in Table 139.

Table 139 – CAPACITY attribute

Usage	Attribute	Description
m	unsigned_integer	Unsigned integer constant greater than or equal to one.

7.22.2.3 COUNT

Purpose

The COUNT attribute specifies the numbers of elements currently stored in the list. When a LIST is part of a FILE, the COUNT shall not be specified since only the EDD application knows how many elements are currently in the list.

If COUNT is specified, it defines the size that it has at creation time of the LIST instance. If it is not defined, a LIST instance with zero elements is created.

Lexical structure

COUNT (unsigned_integer)<exp>

The attribute of COUNT is specified in Table 140.

Table 140 – COUNT attribute

Usage	Attribute	Description
m	unsigned_integer	Unsigned integer constant.

7.23 MENU

7.23.1 General structure

Purpose

MENU organizes variables, methods, and other items specified in the EDDL into a hierarchical structure. A user application may use the MENU ITEMS to display and enter information in an organized and consistent fashion.

Lexical structure

MENU identifier [ITEMS, LABEL, ACCESS, HELP, EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, STYLE, VALIDITY, VISIBILITY]

The attributes of MENU are specified in Table 141.

Table 141 – MENU attributes

Usage	Attribute	Description
m	ITEMS	Lexical element (see 7.23.2.1).
m	LABEL	Lexical element (see 7.36.9).
o	ACCESS	Lexical element (see 7.23.2.2).
o	HELP	Lexical element (see 7.36.8).
o	EXIT_ACTIONS	Lexical element (see 7.23.2.12).
o	INIT_ACTIONS	Lexical element (see 7.23.2.13).
o	POST_EDIT_ACTIONS	Lexical element (see 7.23.2.3).
o	POST_READ_ACTIONS	Lexical element (see 7.23.2.4).
o	POST_WRITE_ACTIONS	Lexical element (see 7.23.2.5).
o	PRE_EDIT_ACTIONS	Lexical element (see 7.23.2.6).
o	PRE_READ_ACTIONS	Lexical element (see 7.23.2.7).
o	PRE_WRITE_ACTIONS	Lexical element (see 7.23.2.8).
o	STYLE	Lexical element (see 7.23.2.11).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).

7.23.2 Specific attributes

7.23.2.1 ITEMS

Purpose

The ITEMS attribute specifies selected elements of the EDD which shall be displayed to the user plus optional qualifiers for each element to control the item processing.

NOTE 1 The MENU items are presented to the user in the order they appear.

NOTE 2 For vertical menus, the first item is displayed on top and the last item on the bottom. For horizontal menus, the first item is displayed on the left and the last item on the right.

COLUMNBREAK and ROWBREAK shall not appear within a conditional.

Lexical structure

ITEMS [(reference, (DISPLAY_VALUE, HIDDEN, READ_ONLY, NO_LABEL, NO_UNIT, REVIEW, INLINE, ALIGN_LEFT, ALIGN_RIGHT))<exp>, SEPARATOR, ROWBREAK, COLUMNBREAK, string] +

The attributes of ITEMS are specified in Table 142.

Table 142 – ITEMS attributes

Usage	Attribute	Description
m	reference	Reference to a VARIABLE, a bit of a BIT_ENUMERATED VARIABLE, a BLOCK_A PARAMETERS, an element of BLOCK_A PARAMETERS, a CHART, a COLLECTION, an EDIT_DISPLAY, a GRAPH, a GRID, an IMAGE, a LIST, a MENU, a METHOD, a METHOD with arguments, a PLUGIN, a RECORD, a REFERENCE_ARRAY, or a VALUE_ARRAY.
c/o	DISPLAY_VALUE	Shall only be used with VARIABLE. Specifies that the value of the corresponding VARIABLE is displayed in addition to its LABEL.
o	HIDDEN	The item is not displayed.
c/o	READ_ONLY	Shall only be used with VARIABLE. Specifies that the value shall be displayed but it may not be modified (regardless of its HANDLING attribute).
c/o	NO_LABEL	Shall only be used with VARIABLE. Specifies that the label of the VARIABLE is not displayed; DISPLAY_VALUE is implied.
c/o	NO_UNIT	Shall only be used with VARIABLE. Specifies that the engineering unit of the VARIABLE is not displayed; DISPLAY_VALUE is implied.
c/o	REVIEW	Shall only be used with MENU. REVIEW is used to present a large quantity of data, for example for review purposes.
c/o	INLINE	Shall only be used with IMAGE. If the IMAGE is in a MENU with a STYLE attribute equal to WINDOW, DIALOG, PAGE or GROUP and the INLINE qualifier is not specified, the IMAGE should span with the width of the MENU; otherwise, the IMAGE should be displayed inline with other ITEMS of the MENU.
c/o	ALIGN_LEFT	Shall only be used with IMAGE. If the IMAGE is in a MENU with a STYLE attribute equal to WINDOW, DIALOG, PAGE or GROUP and the ALIGN_LEFT qualifier is specified, the IMAGE should be displayed left aligned. If no alignment is specified, the IMAGE should be displayed centered.
c/o	ALIGN_RIGHT	Shall only be used with IMAGE. If the IMAGE is in a MENU with a STYLE attribute equal to WINDOW, DIALOG, PAGE or GROUP and the ALIGN_RIGHT qualifier is specified, the IMAGE should be displayed right aligned. If no alignment is specified, the IMAGE should be displayed centered.
c/o	SEPARATOR	Should only be used within a MENU whose STYLE attribute is MENU. The menu items before and after the SEPARATOR should be separated, for example, via a horizontal line.

Usage	Attribute	Description
c/o	COLUMNBREAK	Shall only be used within a MENU whose STYLE attribute is WINDOW, DIALOG, PAGE, or GROUP; specifies a new column should be started.
c/o	ROWBREAK	Shall only be used within a MENU whose STYLE attribute is WINDOW, DIALOG, PAGE, or GROUP; specifies a new row should be started.
o	string	Static text that should be displayed on the menu.

7.23.2.2 ACCESS

Purpose

The ACCESS attribute specifies whether the items displayed or used in the MENU are read from the device (ONLINE) or locally (OFFLINE). If no ACCESS attribute is defined, the ACCESS of the parent MENU shall be applied.

Lexical structure

ACCESS (OFFLINE, ONLINE)

The attributes of ACCESS are specified in Table 143.

Table 143 – ACCESS attribute

Usage	Attribute	Description
s	OFFLINE	The values of the items displayed in the MENU are read locally.
s	ONLINE	The values of the items displayed in the MENU should be read from the device and refreshed with an appropriate interval.

NOTE The implementation is responsible for deciding when the transfer of modified values to a device takes place (typically after a single entry or MENU completion for an ONLINE ACCESS attribute, or via a separate menu to update the device).

7.23.2.3 POST_EDIT_ACTIONS

Purpose

The POST_EDIT_ACTIONS attribute specifies METHODS that shall be executed after the user has finished processing the MENU. If the MENU is aborted, POST_EDIT_ACTIONS are not executed. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

Any POST_EDIT_ACTIONS associated with a referenced entity in the ITEMS shall be executed whenever a new value has been entered for the entity. The POST_EDIT_ACTIONS of the MENU are executed only after all entity specific POST_EDIT_ACTIONS have been completed.

The POST_EDIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

POST_EDIT_ACTIONS [reference]+, DEFINITION

The attributes of POST_EDIT_ACTIONS are specified in Table 144.

Table 144 – EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS attributes

Usage	Attribute	Description
o	reference	Reference to a METHOD instance or a METHOD instance with arguments.
o	DEFINITION	Lexical element (see 7.36.4).

7.23.2.4 POST_READ_ACTIONS

Purpose

The POST_READ_ACTIONS attribute specifies METHODS that shall be executed after all referenced entities in the ITEMS attribute have been read and processed. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed and MENU activation is cancelled.

Any POST_READ_ACTIONS associated with a referenced entity in the ITEMS shall be executed whenever a new value has been read for the entity. The POST_READ_ACTIONS of the MENU shall be executed after all entity specific POST_READ_ACTIONS have been completed.

The POST_READ_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

POST_READ_ACTIONS [reference]+, DEFINITION

The attributes of POST_READ_ACTIONS are specified in Table 144.

7.23.2.5 POST_WRITE_ACTIONS

Purpose

The POST_WRITE_ACTIONS attribute specifies METHODS that shall be executed after all referenced entities in the ITEMS attribute have been written and processed. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

Any POST_WRITE_ACTIONS associated with a referenced entity in the ITEMS shall be executed whenever a new value has been written to the entity. The POST_WRITE_ACTIONS of the MENU shall be executed after all entity specific POST_WRITE_ACTIONS have been completed.

The POST_WRITE_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

POST_WRITE_ACTIONS [reference]+, DEFINITION

The attributes of POST_WRITE_ACTIONS are specified in Table 144.

7.23.2.6 PRE_EDIT_ACTIONS

Purpose

The PRE_EDIT_ACTIONS attribute specifies METHODS that shall be executed immediately after the MENU is activated. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed and MENU activation is cancelled.

The PRE_EDIT_ACTIONS of the MENU shall be executed before any of the defined PRE_EDIT_ACTIONS of the referenced entities. After the MENU PRE_EDIT_ACTIONS are completed, any PRE_EDIT_ACTIONS for the referenced entities shall be executed.

The PRE_EDIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

PRE_EDIT_ACTIONS [reference]+, DEFINITION

The attributes of PRE_EDIT_ACTIONS are specified in Table 144.

7.23.2.7 PRE_READ_ACTIONS

Purpose

The PRE_READ_ACTIONS attribute specifies METHODS that shall be executed before any referenced entities in the ITEMS attribute are read or processed. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed and MENU activation is cancelled. After the MENU PRE_READ_ACTIONS are completed, any PRE_READ_ACTIONS for the referenced entities shall be executed.

The PRE_READ_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

PRE_READ_ACTIONS [reference]+, DEFINITION

The attributes of PRE_READ_ACTIONS are specified in Table 144.

7.23.2.8 PRE_WRITE_ACTIONS

Purpose

The PRE_WRITE_ACTIONS attribute specifies METHODS that shall be executed before any referenced entities in the ITEMS attribute are written or processed. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed. After the MENU PRE_WRITE_ACTIONS are completed, the PRE_WRITE_ACTIONS for the referenced entities shall be executed.

The PRE_WRITE_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

PRE_WRITE_ACTIONS [reference]+, DEFINITION

The attributes of PRE_WRITE_ACTIONS are specified in Table 144.

7.23.2.9 PURPOSE (void)

7.23.2.10 ROLE (void)

7.23.2.11 STYLE

Purpose

The STYLE attribute specifies how the MENU is displayed.

When the STYLE attribute is DIALOG, WINDOW, PAGE or GROUP, the DISPLAY_VALUE qualifier (see 7.23.2.1) is implied on each item on the MENU.

If one or more MENUs are active and a DIALOG MENU is activated, the DIALOG MENU takes precedence and shall be closed before any other MENU can be used.

Lexical structure

STYLE (DIALOG, WINDOW, PAGE, GROUP, MENU, TABLE)

The attributes of STYLE are specified in Table 145.

Table 145 – STYLE attribute

Usage	Attribute	Description
s	DIALOG	No other MENU can be activated while this MENU, or any of its submenus, is active. The MENU shall be closed before interactions with other MENUs are possible, i.e. the items of the MENU should be displayed as a modal dialog box.
s	WINDOW	Other MENUs can be activated while this MENU is active. Interactions with all activated MENUs having WINDOW STYLE attributes are possible, i.e. the items of the MENU should be displayed as a modeless window.
s	PAGE	MENU items should be displayed as a tabbed property page.
s	GROUP	MENU items should be displayed as a group box.
s	MENU	MENU items should be displayed as either a pulldown or popup menu.
s	TABLE	MENU items should be displayed in vertical columns.

7.23.2.12 EXIT_ACTIONS

Purpose

The EXIT_ACTIONS attribute specifies METHODS that shall be executed after the user has finished processing the MENU, even if the MENU is aborted. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

The EXIT_ACTIONS of the MENU shall be executed after all other ACTIONS of the MENU are executed.

The EXIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

EXIT_ACTIONS [reference]+, DEFINITION

The attributes of EXIT_ACTIONS are specified in Table 144.

7.23.2.13 INIT_ACTIONS

Purpose

The INIT_ACTIONS attribute specifies METHODS that shall be executed immediately after the MENU is activated. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed and MENU activation is cancelled.

The INIT_ACTIONS of the MENU shall be executed before any other ACTIONS of the MENU are executed.

The INIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

INIT_ACTIONS [reference]+, DEFINITION

The attributes of INIT_ACTIONS are specified in Table 144.

7.23.3 Sequence diagrams for actions

Figure 30 show the sequence diagrams for actions related to MENUs and their contained VARIABLES and METHODS. Actions are executed and completed in sequence moving down a diagram. If any action fails or aborts for an unplanned reason, the remaining actions are not executed and the MENU is cancelled.

When the MENU has the ACCESS OFFLINE (data coming from a local data base), the pre/post read/write actions for the VARIABLE are not executed.

When the MENU has the ACCESS ONLINE (data is read from or written to the device), the pre/post read/write actions for the VARIABLE shall be executed in the order shown in the sequence.

The init/exit actions and the pre/post edit actions shall always be executed in case of ACCESS OFFLINE and ONLINE.

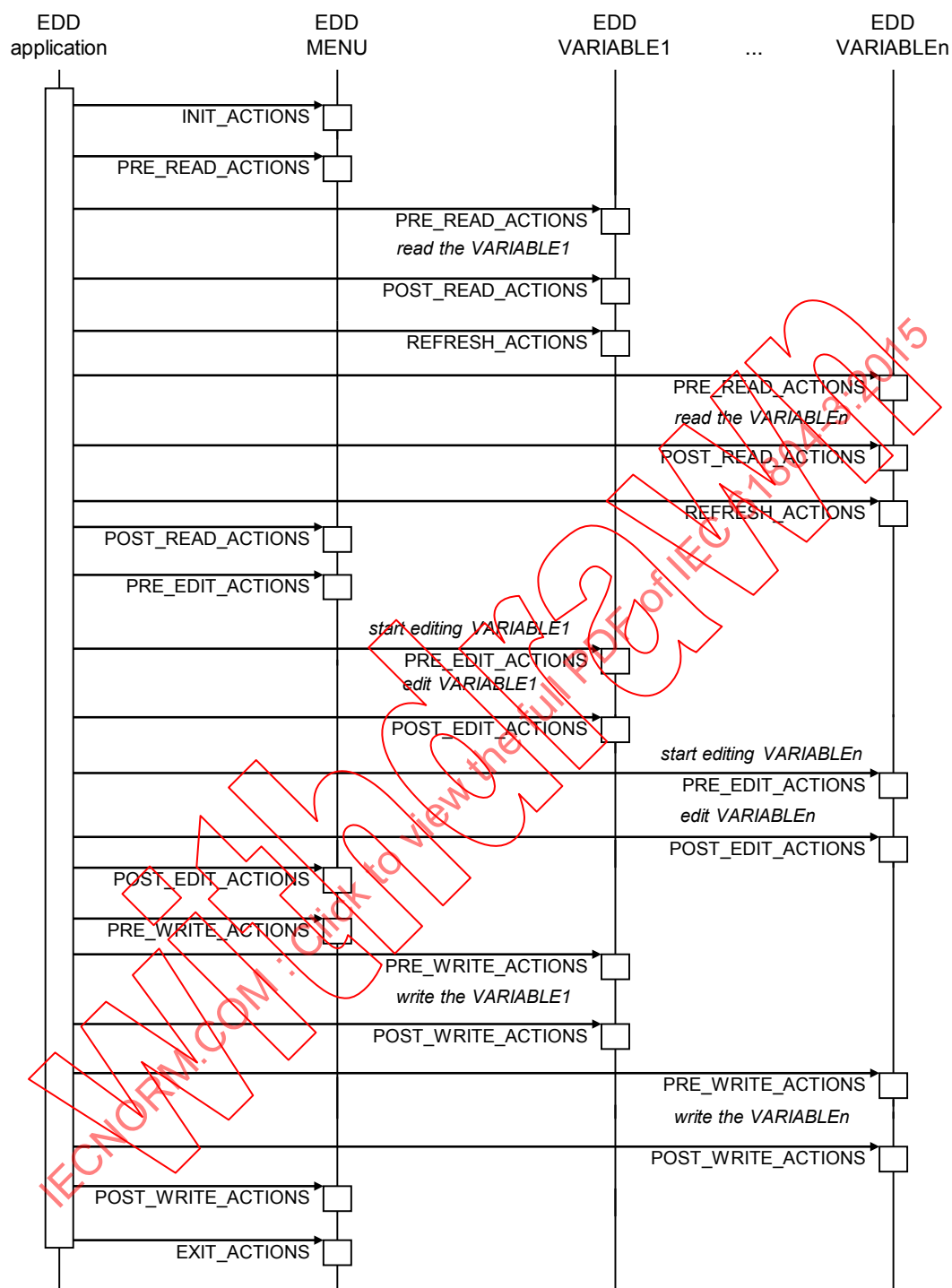


Figure 30 – MENU activation

7.24 METHOD

7.24.1 General structure

Purpose

A METHOD is used to define a subroutine which is executed in the EDD application.

NOTE A METHOD is used to specify consistency checks, conversion algorithms between EDD variables, user acknowledges or specific device interactions.

Lexical structure

METHOD identifier [(parameter-type parameter-name)+] [DEFINITION, ACCESS, CLASS, HELP, LABEL, PRIVATE, TYPE, VALIDITY, VISIBILITY]

The attributes of METHOD are specified in Table 146.

Table 146 – METHOD attributes

Usage	Attribute	Description
m	DEFINITION	Lexical element (see 7.36.4).
o	ACCESS	Lexical element (see 7.24.2.1).
o	CLASS	Lexical element (see 7.24.2.2).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	PRIVATE	Lexical element (see 7.36.18).
o	TYPE	Lexical element (see 7.24.2.3).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	parameter-type	Type of the parameter (see Table 147).
o	parameter-name	Name of the parameter in the form of an identifier.

The possible types of parameters are specified Table 147.

Table 147 – Parameter types

Usage	Attribute	Description
s	char	The METHOD expects a parameter of type char.
s	unsigned char	The METHOD expects a parameter of type unsigned char.
s	short	The METHOD expects a parameter of type short.
s	unsigned short	The METHOD expects a parameter of type unsigned short.
s	int	The METHOD expects a parameter of type int.
s	unsigned int	The METHOD expects a parameter of type unsigned int.
s	long	The METHOD expects a parameter of type long.
s	unsigned long	The METHOD expects a parameter of type unsigned long.
s	long long	The METHOD expects a parameter of type long long.
s	unsigned long long	The METHOD expects a parameter of type unsigned long long.
s	float	The METHOD expects a parameter of type float.
s	double	The METHOD expects a parameter of type double.
s	time_t	The METHOD expects a parameter of type time_t.
s	DD_STRING	The METHOD expects a parameter of type DD_STRING.
s	DD_ITEM	The METHOD expects a parameter of type DD_ITEM.

7.24.2 Specific attributes

7.24.2.1 ACCESS

Purpose

The ACCESS attribute specifies whether the VARIABLE values stored in the device or the offline data set are used within the METHOD DEFINITION. If ACCESS is not defined, the default is ONLINE.

Lexical structure

ACCESS (OFFLINE, ONLINE)

The attributes of ACCESS are specified in Table 148.

Table 148 – ACCESS attributes

Usage	Attribute	Description
s	OFFLINE	The values of the variables used within the METHOD definition are read or written from/to the offline data set.
s	ONLINE	The values of the variables used within the METHOD definition are actually read from the device. Changes of VARIABLE values are written with Builtins.

7.24.2.2 CLASS

Purpose

The CLASS attribute specifies the effect of a METHOD on a device. This attribute is intended to be used by the EDD application to implement permission levels and organize how METHODS are presented.

Lexical structure

CLASS [ALARM, ANALOG_OUTPUT, COMPUTATION, CONTAINED, CORRECTION, DEVICE, DIAGNOSTIC, DISCRETE, FREQUENCY, HART, INPUT, LOCAL_DISPLAY, OPERATE, OUTPUT, SERVICE, SPECIALIST, TUNE] +

The attributes of CLASS are specified in Table 149.

Table 149 – CLASS attributes

Usage	Attribute	Description
s	ALARM	The METHOD is associated with an alarm (e.g., specifying alarm limits, indicating alarm status etc.).
s	ANALOG_OUTPUT	The METHOD is part of an analog output block.
s	COMPUTATION	The METHOD is part of a computation block.
s	CONTAINED	The METHOD represents the physical characteristics of the device.
s	CORRECTION	The METHOD supports the transducers in the device. The METHOD may establish transducer limits, provide signal damping, indicate the transducer's value, linearize or calibrate the transducer, etc.
s	DEVICE	The METHOD specifies the physical characteristics of the device.
s	DIAGNOSTIC	The METHOD evaluates the device status.
s	DISCRETE	The METHOD supports the discrete and or digital I/O of the device.
s	FREQUENCY	The METHOD supports the frequency I/O of the device.
s	HART	The METHOD is part of the HART block which characterizes the HART interface. There is only one HART block.
s	INPUT	The METHOD is part of an input block. An input block is a special kind of computation block which does unit conversions, scaling, and damping. The input block parameters can be determined by the output of another block.
s	LOCAL_DISPLAY	The METHOD is part of the local display block. A local display block contains the VARIABLES associated with the local interface (keyboard, display, etc.) of the device.
s	OPERATE	The METHOD is used to control a block's operation.
s	OUTPUT	The METHOD is part of the output block. The values of output parameters may be accessed by another block input.
s	SERVICE	The METHOD is used when performing routine maintenance.
s	SPECIALIST	The METHOD shall be only executable for a specialist user of the EDD application. For a non-specialist user the METHOD with this CLASS attribute shall be not executable. If VALIDITY or VISIBILITY is FALSE the METHOD shall not be visible. If the attribute is not defined the METHOD shall be executable.
s	TUNE	The METHOD is used to tune the algorithm of a block.

7.24.2.3 TYPE

Purpose

The TYPE attribute specifies the type of the returned value.

Lexical structure

TYPE [char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long, float, double, time_t, DD_STRING]

The attributes of TYPE are specified in Table 150.

Table 150 – TYPE attributes

Usage	Attribute	Description
s	char	The METHOD returns a value of type char.
s	unsigned char	The METHOD returns a value of type unsigned char.
s	short	The METHOD returns a value of type short.
s	unsigned short	The METHOD returns a value of type unsigned short.
s	int	The METHOD returns a value of type int.
s	unsigned int	The METHOD returns a value of type unsigned int.
s	long	The METHOD returns a value of type long.
s	unsigned long	The METHOD returns a value of type unsigned long.
s	long long	The METHOD returns a value of type long long.
s	unsigned long long	The METHOD returns a value of type unsigned long long.
s	float	The METHOD returns a value of type float.
s	double	The METHOD returns a value of type double.
s	time_t	The METHOD returns a value of type time_t.
s	DD_STRING	The METHOD returns a value of type DD_STRING.

7.25 PROGRAM (void)**7.26 RECORD****Purpose**

A RECORD is a logical group of VARIABLES used to represent a complex communication object. Each MEMBER in the RECORD is a reference to a VARIABLE and may be of different data types.

Lexical structure

RECORD identifier [MEMBERS, LABEL, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]

The attributes of RECORD are specified in Table 151.

Table 151 – RECORD attributes

Usage	Attribute	Description
m	MEMBERS	Lexical element (see 7.36.12).
m	LABEL	Lexical element (see 7.36.9).
o	HELP	Lexical element (see 7.36.8).
o	PRIVATE	Lexical element (see 7.36.18).
o	RESPONSE_CODES	Lexical element (see 7.36.14).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	WRITE_MODE	Lexical element (see 7.36.20).

7.27 REFERENCE_ARRAY

7.27.1 General structure

Purpose

A REFERENCE_ARRAY is a set of items of the specified item type (for example, VARIABLE or MENU).

NOTE REFERENCE_ARRAYs are not related to communication arrays (item type "VALUE_ARRAY"). Communication arrays are arrays of values.

Lexical structure

REFERENCE_ARRAY identifier [item-type, ELEMENTS, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]

The attributes of REFERENCE_ARRAY are specified in Table 152.

Table 152 – REFERENCE_ARRAY attributes

Usage	Attribute	Description
m	item-type	Type of items listed within the ELEMENTS attribute. Shall be one of the basic elements listed in Table 50.
m	ELEMENTS	Lexical element (see 7.27.2).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	PRIVATE	Lexical element (see 7.36.15).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).

7.27.2 Specific attribute – ELEMENTS

Purpose

The ELEMENTS attribute specifies a list of items. It defines for each item the item reference, its associated index, optional description and help. The description and help attributes, when used, override the corresponding specifications in the EDD item itself.

Lexical structure

ELEMENTS [integer, reference, description, help]<exp>]+

The attributes of ELEMENTS are specified in Table 153.

Table 153 – ELEMENTS attributes

Usage	Attribute	Description
m	integer	Number by which the item may be referenced. Each number shall be uniquely defined within a REFERENCE_ARRAY construct.
m	reference	Reference to a data-item instance.
o	description	Short description for the item.
o	help	Help text for the item.

7.28 Relations

7.28.1 REFRESH

Purpose

Writing a parameter to a device may cause the device to update the values of other parameters. A REFRESH relation defines the dependency between the cause-parameter set and the affected-parameter set. The affected-parameter set shall be read if any value of the cause-parameter set is modified, i.e set to a new value. The modification may be initiated by user interactions inside the EDD application, inside METHOD executions or by reading a new value from the device.

A VARIABLE may have a refresh relationship with itself, implying that the VARIABLE is to be read after writing.

A VARIABLE of the affected-parameter set may be used in another REFRESH relation as a cause-parameter.

The REFRESH relation shall not be executed when the EDD application performs a download (see IEC 61804-4).

Lexical structure

REFRESH identifier, [(cause-parameter)<exp>]+, [(affected-parameter)<exp>]+

The attributes of REFRESH are specified in Table 154.

Table 154 – REFRESH attributes

Usage	Attribute	Description
m	cause-parameter	Set of VARIABLE instances, RECORDs or VALUE_ARRAYs.
m	affected-parameter	Set of VARIABLE instances, LISTs, RECORDs or VALUE_ARRAYs, which shall be read, if one of the cause parameters has been modified. A VARIABLE of the affected parameter set may also be in the cause-parameter set of another REFRESH relation.

7.28.2 UNIT

Purpose

A UNIT relation specifies a reference to a VARIABLE holding a unit and a list of dependent items. When the VARIABLE holding the unit is modified, the list of affected items using that unit shall be refreshed. When an affected item is displayed, its unit shall also be displayed.

Lexical structure

UNIT identifier, [cause-parameter)<exp>], [(affected-parameter)<exp>]+

The attributes of UNIT are specified in Table 155.

Table 155 – UNIT attributes

Usage	Attribute	Description
m	cause-parameter	Reference to an enumerated typed VARIABLE instance containing the unit code or to a string typed VARIABLE instance containing the unit string.
m	affected-parameter	List of VARIABLE, VALUE_ARRAY, LIST or AXIS instances using that unit code.

7.28.3 WRITE_AS_ONE

Purpose

A WRITE_AS_ONE relation specifies a list of all data items that shall be visible at the time of editing any of the data items. The EDD application shall display all elements of the relation, even if only one element of this relation is referenced in a MENU or EDIT_DISPLAY.

A WRITE_AS_ONE relation shall only be used when a device requires specific data items to be examined and modified at the same time for proper operation. If the variables could be modified independently, a WRITE_AS_ONE relation shall not be used. In order to ensure consistent layout, all data items should be referenced on the MENU or EDIT_DISPLAY.

The WRITE_AS_ONE relation only affects the user interface of the EDD application and not the communication behavior. The relation does not force the EDD application to write all of the elements if only some of the elements have been changed.

Lexical structure

WRITE_AS_ONE identifier, [(reference)<exp>]+

The attribute of WRITE_AS_ONE is specified in Table 156.

Table 156 – WRITE_AS_ONE attribute

Usage	Attribute	Description
m	reference	List of VARIABLES, VALUE_ARRAYs and RECORDs instances that shall be displayed.

7.29 RESPONSE_CODES

Purpose

RESPONSE_CODES specify values a device may return as error information. Each VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST or COMMAND can have its own set of RESPONSE_CODES.

Lexical structure

RESPONSE_CODES identifier, [(integer, [DATA_ENTRY_ERROR, DATA_ENTRY_WARNING, MISC_ERROR, MISC_WARNING, MODE_ERROR, PROCESS_ERROR, SUCCESS], description, help*)<exp>]+

The attributes of RESPONSE_CODES are specified in Table 157.

Table 157 – RESPONSE_CODES attributes

Usage	Attribute	Description
m	description	String that will be displayed when the response code is returned by the device.
m	integer	The returned value of the device to identify the response code.
s	DATA_ENTRY_ERROR	The application layer service was rejected because the data sent was invalid.
s	DATA_ENTRY_WARNING	The application layer service was accepted and processed with a slightly modified version of the data sent.
s	MISC_ERROR	The application layer service was rejected.
s	MISC_WARNING	The application layer service was accepted and processed as specified and there is additional information, unrelated to the application layer service, in which the user might be interested.
s	MODE_ERROR	The application layer service was rejected because the device was in a mode in which the application layer service could not be executed.
s	PROCESS_ERROR	The application layer service was rejected because a process applied to the device was invalid.
s	SUCCESS	The application layer service was accepted and processed as specified.
o	help	Help text for the item.

7.30 SOURCE

7.30.1 General structure

Purpose

A SOURCE describes a source of data values for a CHART.

Lexical structure

SOURCE identifier [MEMBERS, EMPHASIS, EXIT_ACTIONS, HELP, INIT_ACTIONS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY, VISIBILITY, Y_AXIS]

The attributes of SOURCE are specified in Table 158.

Table 158 – SOURCE attributes

Usage	Attribute	Description
m	MEMBERS	Lexical element (see 7.36.12).
o	EMPHASIS	Lexical element (see 7.36.5).
o	EXIT_ACTIONS	Lexical element (see 7.30.2.1).
o	HELP	Lexical element (see 7.36.8).
o	INIT_ACTIONS	Lexical element (see 7.30.2.2).
o	LABEL	Lexical element (see 7.36.9).
o	LINE_COLOR	Lexical element (see 7.36.10).
o	LINE_TYPE	Lexical element (see 7.36.11).
o	REFRESH_ACTIONS	Lexical element (see 7.30.2.3).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	Y_AXIS	Lexical element (see 7.30.2.4).

For each SOURCE not specifying a LINE_TYPE or LINE_COLOR, each line shall have a different appearance from the others on the CHART.

7.30.2 Specific attributes

7.30.2.1 EXIT_ACTIONS

Purpose

The EXIT_ACTIONS attribute specifies actions that are executed when the CHART containing the SOURCE is closed.

The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the remaining METHODS shall not be executed.

The EXIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

EXIT_ACTIONS [reference]+, DEFINITION

The attributes of EXIT_ACTIONS are specified in Table 188.

7.30.2.2 INIT_ACTIONS

Purpose

The INIT_ACTIONS attribute specifies actions that are executed before the SOURCE value is initially displayed by a CHART.

The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the remaining METHODS shall not be executed. Furthermore, the EDD application shall not draw the SOURCE value if a METHOD exits for an unplanned reason.

The INIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

INIT_ACTIONS [reference]+, DEFINITION

The attributes of INIT_ACTIONS are specified in Table 188.

7.30.2.3 REFRESH_ACTIONS

Purpose

The REFRESH_ACTIONS attribute specifies actions that are executed prior to the display of the value on the CHART in which the SOURCE is being displayed.

The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the remaining METHODS shall not be executed.

The REFRESH_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

REFRESH_ACTIONS [reference]+, DEFINITION

The attributes of REFRESH_ACTIONS are specified in Table 188.

7.30.2.4 Y_AXIS

Purpose

The Y_AXIS attribute specifies the way the Y-axis of the SOURCE is displayed. Each SOURCE on a CHART shares a common X-axis that is based on time, but each SOURCE on a CHART has its own Y_AXIS.

Lexical structure

Y_AXIS (reference)<exp>

The attribute of Y_AXIS is specified in Table 159.

Table 159 – Y_AXIS attribute

Usage	Attribute	Description
m	reference	Reference to an AXIS instance.

7.31 TEMPLATE

7.31.1 General structure

Purpose

TEMPLATE describes a set of parameter values that is used for offline configuration by the EDD application. Each TEMPLATE defines default values for application specific configuration. The DEFAULT_VALUES attribute defines the default value of parameters.

Lexical structure

TEMPLATE identifier [DEFAULT_VALUES, HELP, LABEL, VALIDITY]

The attributes of TEMPLATE are specified in Table 160.

Table 160 – TEMPLATE attributes

Usage	Attribute	Description
m	DEFAULT_VALUES	Lexical element (see 7.31.2).
m	LABEL	Lexical element (see 7.36.9).
o	HELP	Lexical element (see 7.36.8).
o	VALIDITY	Lexical element (see 7.36.16).

7.31.2 Specific attribute – DEFAULT_VALUES

Purpose

The DEFAULT_VALUES attribute specifies the default settings for VARIABLES, VALUE_ARRAYs or LISTs. The values to be set shall be constants.

Lexical structure

DEFAULT_VALUES [reference, (value*, (value+)*)]+

The attributes of DEFAULT_VALUES are specified in Table 161.

Table 161 – DEFAULT_VALUES attributes

Usage	Attribute	Description
m	reference	Reference to a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance, a VALUE_ARRAY instance or a LIST instance.
c/m	value	For a VARIABLE a constant that specifies the value. The type depends on the data type specified with the reference attribute. For a VALUE_ARRAY or a LIST, a set of values. May be nested for a multidimensional VALUE_ARRAY or LIST.

EDDs with BLOCK_A shall reference data items using block referencing (see 7.38).

7.32 VALUE_ARRAY

7.32.1 General structure

Purpose

A VALUE_ARRAY is a logical group of values. Each element of a VALUE_ARRAY shall be of the same data type and shall correspond to an EDDL construct. An element is referenced from elsewhere in the EDD via the VALUE_ARRAY identifier and the element index.

NOTE From a communication perspective, the individual elements of the VALUE_ARRAY are not treated as individual variables but simply as a grouped value.

Lexical structure

VALUE_ARRAY identifier [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]

The attributes of VALUE_ARRAY are specified in Table 162.

Table 162 – VALUE_ARRAY attributes

Usage	Attribute	Description
m	LABEL	Lexical element (see 7.36.9).
m	NUMBER_OF_ELEMENTS	Lexical element (see 7.32.2.1).
m	TYPE	Lexical element (see 7.32.2.2).
o	HELP	Lexical element (see 7.36.8).
o	PRIVATE	Lexical element (see 7.36.18).
o	RESPONSE_CODES	Lexical element (see 7.36.14).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	WRITE_MODE	Lexical element (see 7.36.20).

7.32.2 Specific attributes

7.32.2.1 NUMBER_OF_ELEMENTS

Purpose

The `NUMBER_OF_ELEMENTS` attribute specifies the number of elements in the `VALUE_ARRAY` as an integer greater than zero.

Lexical structure

`NUMBER_OF_ELEMENTS` (integer, expression)

The attributes of `NUMBER_OF_ELEMENTS` are specified in Table 163.

Table 163 – `NUMBER_OF_ELEMENTS` attributes

Usage	Attribute	Description
s	integer	Number of elements of a <code>VALUE_ARRAY</code> .
s	expression	Expression which shall be evaluated to a non-negative integer that is within the index range of the <code>NUMBER_OF_ELEMENTS</code> attribute; for the description of expressions, see 7.40.

7.32.2.2 TYPE

Purpose

The `TYPE` attribute is a reference to a `VARIABLE`, `COLLECTION`, `LIST`, `VALUE_ARRAY` or `REFERENCE_ARRAY`. All attributes from the EDDL construct apply to each of the `VALUE_ARRAY` elements.

Lexical structure

`TYPE` reference

The `TYPE` attribute is specified in Table 164.

Table 164 – `TYPE` attribute

Usage	Attribute	Description
m	reference	Reference to an EDD basic construct instance. The reference shall terminate to a data item. The item referenced in the <code>TYPE</code> attribute of a <code>VALUE_ARRAY</code> shall not contain <code>VALIDITY</code> or any other conditional expressions that would cause data type structures to change. This restriction also applies for contained items.

7.33 VARIABLE

7.33.1 General structure

Purpose

`VARIABLE` describes parameters contained in a device or in the EDD application.

Lexical structure

`VARIABLE` identifier, [`CLASS`, `LABEL`, `TYPE`, `CONSTANT_UNIT`, `DEFAULT_VALUE`, `HANDLING`, `HEIGHT`, `HELP`, `INITIAL_VALUE`, `POST_EDIT_ACTIONS`, `POST_READ_ACTIONS`, `POST_RQSTUPDATE_ACTIONS`, `POST_USERCHANGE_ACTIONS`, `POST_WRITE_ACTIONS`, `PRE_EDIT_ACTIONS`, `PRE_READ_ACTIONS`, `PRE_WRITE_ACTIONS`, `PRIVATE`, `REFRESH_ACTIONS`, `RESPONSE_CODES`, `VALIDITY`, `VISIBILITY`, `WIDTH`, `WRITE_MODE`]

The attributes of `VARIABLE` are specified in Table 165.

Table 165 – VARIABLE attributes

Usage	Attribute	Description
m	CLASS	Lexical element (see 7.33.2.1).
o	LABEL	Lexical element (see 7.36.9).
m	TYPE	Lexical element (see 7.33.2.2).
o	CONSTANT_UNIT	Lexical element (see 7.33.2.3).
o	DEFAULT_VALUE	Lexical element (see 7.33.2.4).
o	HANDLING	Lexical element (see 7.36.6).
o	HEIGHT	Lexical element (see 7.36.7).
o	HELP	Lexical element (see 7.36.8).
o	INITIAL_VALUE	Lexical element (see 7.33.2.5).
o	POST_EDIT_ACTIONS	Lexical element (see 7.33.2.6).
o	POST_READ_ACTIONS	Lexical element (see 7.33.2.7).
o	POST_RQSTUPDATE_ACTIONS	Lexical element (see 7.33.2.15).
o	POST_USERCHANGE_ACTIONS	Lexical element (see 7.33.2.16).
o	POST_WRITE_ACTIONS	Lexical element (see 7.33.2.8).
o	PRE_EDIT_ACTIONS	Lexical element (see 7.33.2.9).
o	PRE_READ_ACTIONS	Lexical element (see 7.33.2.10).
o	PRE_WRITE_ACTIONS	Lexical element (see 7.33.2.11).
o	PRIVATE	Lexical element (see 7.36.18).
o	REFRESH_ACTIONS	Lexical element (see 7.33.2.13).
o	RESPONSE_CODES	Lexical element (see 7.36.14).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	WIDTH	Lexical element (see 7.36.17).
o	WRITE_MODE	Lexical element (see 7.36.20).

7.33.2 Specific attributes

7.33.2.1 CLASS

Purpose

The CLASS attribute specifies how the VARIABLE is used by the device and the EDD application for organization purposes and display.

Not all class combinations are allowed. The restrictions are defined in the profile tables (see Annex D).

The definition of CLASS is based on a model in which a device is made up of a number of different functional blocks. Each block has an input and produces an output. Blocks can be combined in different ways to produce different functionalities.

Lexical structure

CLASS [ALARM, ANALOG_INPUT, ANALOG_OUTPUT, COMPUTATION, CONTAINED, CORRECTION, DEVICE, DIAGNOSTIC, DIGITAL_INPUT, DIGITAL_OUTPUT, DISCRETE_INPUT, DISCRETE_OUTPUT, DYNAMIC, FREQUENCY_INPUT, FREQUENCY_OUTPUT, HART, INPUT, IS_CONFIG, LOCAL, LOCAL_DISPLAY, OPERATE, OPTIONAL, OUTPUT, SERVICE, SPECIALIST, TEMPORARY, TUNE]+

The attributes of CLASS are specified in Table 166.

Table 166 – CLASS attributes

Usage	Attribute	Description
s	ALARM	The VARIABLE contains alarm limits.
s	ANALOG_INPUT	The VARIABLE is part of an analog input block.
s	ANALOG_OUTPUT	The VARIABLE is part of an analog output block.
s	COMPUTATION	The VARIABLE is part of a computation block.
s	CONTAINED	The VARIABLE represents the physical characteristics of the device.
s	CORRECTION	The VARIABLE is part of the correction block.
s	DEVICE	The VARIABLE represents the physical characteristics of the device.
s	DIAGNOSTIC	The VARIABLE indicates the device status.
s	DIGITAL_INPUT	The VARIABLE is part of a digital input block.
s	DIGITAL_OUTPUT	The VARIABLE is part of a digital output block.
s	DISCRETE_INPUT	The VARIABLE is part of a discrete input block.
s	DISCRETE_OUTPUT	The VARIABLE is part of a discrete output block.
s	DYNAMIC	The VARIABLE is modified by the device without stimulus from the network.
s	FREQUENCY_INPUT	The VARIABLE is part of a frequency input block.
s	FREQUENCY_OUTPUT	The VARIABLE is part of a frequency output block.
s	HART	The VARIABLE is part of the HART block which characterizes the HART interface. There is only one HART block.
s	INPUT	The VARIABLE is part of an input block. An input block is a special kind of computation block which does unit conversions, scaling, and damping. The parameter of the input block parameters can be determined by the output of another block.
s	IS_CONFIG	A modification of the VARIABLE results in setting the revision counter or configuration-changed bit, as applicable.
s	LOCAL	The VARIABLE is locally used by the EDD application. Local VARIABLES are not stored in a device, but they can be sent to a device.
s	LOCAL_DISPLAY	The VARIABLE is part of the local display block. A local display block contains the VARIABLES associated with the local interface (keyboard, display, etc.) of the device.
s	OPERATE	The VARIABLE is used to control a block's operation.
s	OPTIONAL	The VARIABLE may not be present in the device.
s	OUTPUT	The VARIABLE is part of the output block. The values of output VARIABLES may be accessed by another block input.
s	SERVICE	The VARIABLE is used when performing routine maintenance.
s	SPECIALIST	The VARIABLE shall only be editable if a specialist is using the EDD application. For a non-specialist user the VARIABLE with this CLASS attribute shall be read only independently of the specified HANDLING. If VALIDITY or VISIBILITY is FALSE the VARIABLE shall not be visible. If the attribute is not defined the VARIABLE shall be editable.
s	TEMPORARY	The VARIABLE is initialized to its DEFAULT_VALUE in each EDD application session. LOCAL variables may be persisted by the EDD application; TEMPORARY variables are never persisted.
s	TUNE	The VARIABLE is used to tune the algorithm of a block.

7.33.2.2 TYPE

7.33.2.2.1 General structure

Purpose

The TYPE attribute describes the format of the VARIABLES value. The column “Initial value” in Table D.9 shows the initial values in case TYPE is specified without DEFAULT_VALUE and INITIAL_VALUE.

Lexical structure

TYPE [DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER, DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE, BIT_ENUMERATED, ENUMERATED, INDEX, OBJECT_REFERENCE, ASCII, BITSTRING, EUC, OCTET, PACKED_ASCII, PASSWORD, VISIBLE, BOOLEAN]

The attributes of TYPE are specified in Table 167.

Table 167 – TYPE attributes

Usage	Attribute	Description
s	BOOLEAN	Boolean type (see 7.33.2.2.2.8).
s	DOUBLE	Arithmetic type (see 7.33.2.2.2.1).
s	FLOAT	Arithmetic type (see 7.33.2.2.2.1).
s	INTEGER	Arithmetic type (see 7.33.2.2.2.1).
s	UNSIGNED_INTEGER	Arithmetic type (see 7.33.2.2.2.1).
s	DATE	Date/time type (see 7.33.2.2.2.2).
s	DATE_AND_TIME	Date/time type (see 7.33.2.2.2.2).
s	DURATION	Date/time type (see 7.33.2.2.2.2).
s	TIME	Date/time type (see 7.33.2.2.2.2).
s	TIME_VALUE(8)	Date/time type (see 7.33.2.2.2.2).
s	TIME_VALUE(4)	Date/time type (see 7.33.2.2.2.2).
s	BIT_ENUMERATED	Enumeration type (see 7.33.2.2.2.3).
s	ENUMERATED	Enumeration type (see 7.33.2.2.2.4).
s	INDEX	Index type (see 7.33.2.2.2.5).
s	OBJECT_REFERENCE	Object reference (see 7.33.2.2.2.6).
s	ASCII	String type (see 7.33.2.2.2.7).
s	BITSTRING	String type (see 7.33.2.2.2.7).
s	EUC	String type (see 7.33.2.2.2.7).
s	OCTET	String type (see 7.33.2.2.2.7).
s	PACKED_ASCII	String type (see 7.33.2.2.2.7).
s	PASSWORD	String type (see 7.33.2.2.2.7).
s	VISIBLE	String type (see 7.33.2.2.2.7).

7.33.2.2.2 Specific attributes

7.33.2.2.2.1 DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Purpose

VARIABLEs of TYPE FLOAT and DOUBLE are single-precision basic format and double-precision basic format floating-point numbers, as defined in IEEE 754, respectively. VARIABLEs of TYPE INTEGER and UNSIGNED_INTEGER are signed and unsigned integer numbers, respectively.

These types are known collectively as the arithmetic types.

Lexical structure — DOUBLE

DOUBLE [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR], [(description, help, value)<exp>]*

Lexical structure — FLOAT

FLOAT [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR], [(description, help, value)<exp>]*

Lexical structure — INTEGER

INTEGER [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR, size], [description, help, value]*

Lexical structure — UNSIGNED_INTEGER

UNSIGNED_INTEGER [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR, size], [(description, help, value)<exp>]*

The attributes of the arithmetic types are specified in Table 168.

Table 168 – DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER attributes

Usage	Attribute	Description
o	DEFAULT_VALUE	Specifies the default setting for the VARIABLE. The DEFAULT_VALUE is available for all EDDL types. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as an INITIAL_VALUE. The DEFAULT_VALUE can also be built by an expression (see 7.40). Lexical structure DEFAULT_VALUE (double)<exp> DEFAULT_VALUE (float)<exp> DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	description	A short description of the item.
o	DISPLAY_FORMAT	Specifies how the value of the VARIABLE is displayed. The DISPLAY_FORMAT string shall be defined according to the functions printf and scanf of the programming language-C (see ISO/IEC 9899). Lexical structure DISPLAY_FORMAT (string)<exp>
o	EDIT_FORMAT	Specifies the format for the editing. The EDIT_FORMAT string shall be defined according to the functions printf and scanf of the programming language-C (see ISO/IEC 9899). Lexical structure EDIT_FORMAT (string)<exp>

Usage	Attribute	Description
o	help	Specifies help text for the item.
o	INITIAL_VALUE	When a device is instantiated for the first time, the INITIAL_VALUE of a VARIABLE is displayed. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned. Lexical structure INITIAL_VALUE (double) INITIAL_VALUE (float) INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)
o	MAX_VALUE	Specifies the upper bound of values to which a VARIABLE shall be set. If the VARIABLE is CLASS DYNAMIC, the device should set the value of the VARIABLE inside the range specified by its minimum and maximum values. A VARIABLE may have more than one maximum value, for instance, to specify more than one range of validity. When there are multiple maximum values, an additional integer is used to number each MAX_VALUE. Every MAX_VALUE shall have an equivalent MIN_VALUE. The range specified with a pair of MIN_VALUE and MAX_VALUE may not overlap another specified range. If a MIN_VALUE/MAX_VALUE has no equivalent MAX_VALUE/MIN_VALUE, the upper/lower bound is unlimited and need not be specified. The MAX_VALUE can also be built by an expression (see 7.40). Lexical structure MAX_VALUE (double, integer)<exp> MAX_VALUE (float, integer)<exp> MAX_VALUE (integer, integer)<exp> MAX_VALUE (unsigned_integer, integer)<exp> MAX_VALUE (expression)<exp>
c	m	An increasing integer starting with one, which identifies the MAX_VALUE if more than one is present.
o	MIN_VALUE	Specifies the lower bound of values to which a VARIABLE shall be set. If the VARIABLE is CLASS DYNAMIC, the device should set the value of the VARIABLE inside the range specified by its minimum and maximum values. A VARIABLE may have more than one minimum value, for instance, to specify more than one range of validity. When there are multiple minimum values, an additional integer is used to number each MIN_VALUE. Every MIN_VALUE shall have an equivalent MAX_VALUE. The range specified with a pair of MIN_VALUE and MAX_VALUE may not overlap another specified range. If a MIN_VALUE/MAX_VALUE has no equivalent MAX_VALUE/MIN_VALUE, the upper/lower bound is unlimited and need not be specified. The MIN_VALUE can also be built by an expression (see 7.40). Lexical structure MIN_VALUE (double, integer)<exp> MIN_VALUE (float, integer)<exp> MIN_VALUE (integer, integer)<exp> MIN_VALUE (unsigned_integer, integer)<exp> MIN_VALUE (expression)<exp>
c	n	An increasing integer starting with one, which identifies the MIN_VALUE if more than one is present.

Usage	Attribute	Description
o	SCALING_FACTOR	Indicates that the displayed value of the VARIABLE is not the value returned by a device. The displayed value is the value returned by a device multiplied by a factor. Therefore, the EDDL processor shall multiply the value of the VARIABLE returned by a device with its SCALING_FACTOR before it is displayed (or before it is used in any other way). The SCALING_FACTOR shall not equal zero. The SCALING_FACTOR can also be built by an expression (see 7.40). The SCALING_FACTOR is applied for display purposes only; the value of the variable, when assigned to a method local variable, is the unscaled value returned by the device; minimum and maximum values, when used in conjunction with the scaling factor, are also unscaled values. Lexical structure SCALING_FACTOR (double)<exp> SCALING_FACTOR (expression)<exp>
o	size	Size of the data type in octets. Size is an integer constant greater than zero and has no upper bound. This value is optional. The default is 1.
o	value	Constant that specifies the value. The type depends on the data type specified with the TYPE attribute which is optional, but if defined a description attribute is required. Equal values are not allowed.

7.33.2.2.2.2 DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

Purpose

VARIABLEs of data type DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE are used to specify dates, times, dates and times, and time differences:

- the data type DATE consists of a calendar date;
- the data type DATE_AND_TIME consists of a calendar date and a time;
- the data type DURATION is a time difference which consists of a time in ms and a day count;
- the data type TIME consists of a time and an optional date;
- the data type TIME_VALUE is used to represent time differences, time of day, or absolute time values in the required precision for application clock synchronization.

Lexical structure – DATE

DATE specifies that VARIABLE is an integer of three octets. The coding of the octets is specified in D.6.2.2.

DATE [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Lexical structure – DATE_AND_TIME

DATE_AND_TIME specifies that VARIABLE is an integer of seven octets. The coding of the octets is specified in D.6.2.3.

DATE_AND_TIME [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Lexical structure – DURATION

DURATION specifies that VARIABLE is an integer of 6 octets. The first four bits shall have the value zero. Bit 0 to 27 represents the time in ms. The number of days is a 16-bit binary value. The coding of the octets is specified in D.6.2.4.

DURATION [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Lexical structure — TIME

TIME specifies that VARIABLE is an integer of six octets. The time is stated in ms starting at midnight. At midnight, time counting starts with zero.

The date is stated in days relatively to 1984-01-01. On the first day of January 1984, date counting starts with the value zero. The coding of the octets is specified in D.6.2.5.

TIME [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Lexical structure – TIME_VALUE

TIME_VALUE specifies that VARIABLE is an integer of four or eight octets. It is a fixed point number, which is the time in multiples of 1/32 ms. The coding of the octets is specified in D.6.2.6.

TIME_VALUE [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*, size, TIME_FORMAT, TIME_SCALE]

The attributes of the DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE types are specified in Table 169.

Table 169 – DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE attributes

Usage	Attribute	Description
o	DEFAULT_VALUE	Specifies the default setting for the VARIABLE. The DEFAULT_VALUE is available for all EDD types. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as an INITIAL_VALUE. The DEFAULT_VALUE can also be built by an expression (see 7.40). Lexical structure DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	DISPLAY_FORMAT	See Table 168.
o	EDIT_FORMAT	See Table 168.
o	INITIAL_VALUE	When a device is instantiated for the first time, the INITIAL_VALUE of a VARIABLE is displayed. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned. Lexical structure INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)
o	MAX_VALUE	Specifies the upper bound of values to which a VARIABLE shall be set. If the VARIABLE is CLASS DYNAMIC, the device should set the value of the VARIABLE inside the range specified by its minimum and maximum values. A VARIABLE may have more than one maximum value, for instance, to specify more than one range of validity. When there are multiple maximum values, an additional integer is used to number each MAX_VALUE. Every MAX_VALUE shall have an equivalent MIN_VALUE. The range specified with a pair of MIN_VALUE and MAX_VALUE may not overlap another specified range. If a MIN_VALUE/MAX_VALUE has no equivalent MAX_VALUE/MIN_VALUE, the upper/lower bound is unlimited and need not be specified. The MAX_VALUE can also be built by an expression (see 7.40). Lexical structure MAX_VALUE (date)<exp> MAX_VALUE (date_and_time)<exp> MAX_VALUE (duration)<exp> MAX_VALUE (time)<exp> MAX_VALUE (time_value)<exp> MAX_VALUE (expression)<exp>

Usage	Attribute	Description
c	m	An increasing integer starting with one, which identifies the MAX_VALUE if more than one is present.
o	MIN_VALUE	Specifies the lower bound of values to which a VARIABLE shall be set. If the VARIABLE is CLASS DYNAMIC, the device should set the value of the VARIABLE inside the range specified by its minimum and maximum values. A VARIABLE may have more than one minimum value, for instance, to specify more than one range of validity. When there are multiple minimum values, an additional integer is used to number each MIN_VALUE. Every MIN_VALUE shall have an equivalent MAX_VALUE. The range specified with a pair of MIN_VALUE and MAX_VALUE may not overlap another specified range. If a MIN_VALUE/MAX_VALUE has no equivalent MAX_VALUE/MIN_VALUE, the upper/lower bound is unlimited and need not be specified. The MIN_VALUE can also be built by an expression (see 7.40). Lexical structure MIN_VALUE (date)<exp> MIN_VALUE (date_and_time)<exp> MIN_VALUE (duration)<exp> MIN_VALUE (time)<exp> MIN_VALUE (time_value)<exp> MIN_VALUE (expression)<exp>
c	n	An increasing integer starting with one, which identifies the MIN_VALUE if more than one is present.
o	size	Size of the data type in octets. Shall be either 4 or 8. The default is 4. This attribute is only applicable to TIME_VALUE.
o	TIME_FORMAT	Specifies how the VARIABLE is displayed and edited. The TIME_FORMAT strings may include the %H, %I, %M, %p, %R, %S, and %T format specifiers of the programming language-C function strftime. For DATE_AND_TIME and TIME_VALUE(8), the following additional time format specifiers apply: %a %A %b %B %c %C %d %D %e %F %g %G %h %j %m %u %U %V %w %W %x %y %Y. When TIME_FORMAT is specified, DISPLAY_FORMAT and EDIT_FORMAT shall not be specified. Lexical structure TIME_FORMAT (string)<exp>
o	TIME_SCALE	Specifies how the VARIABLE is scaled, which may be SECONDS, MINUTES, or HOURS. When present, TIME_SCALE infers a scaling factor of 1/32 000, 1/1 920 000, or 1/115 200 000 and a constant unit of seconds ("s"), minutes ("min"), or hours ("h"), respectively. When TIME_SCALE is specified, DISPLAY_FORMAT and EDIT_FORMAT may also be specified. This attribute is only applicable to TIME_VALUE(4). Lexical structure TIME_SCALE (SECONDS, MINUTES, HOURS)<exp>

7.33.2.2.2.3 BIT_ENUMERATED

Purpose

A VARIABLE of type BIT_ENUMERATED is an unsigned integer defined as a set of single bit definitions. Each bit is identified by the value of its bit position. Bit 0 is identified by value 1 (2^0), bit 1 by value 2 (2^1), bit 2 by value 4 (2^2), bit 3 by value 8 (2^3), and so on.

Lexical structure

```
BIT_ENUMERATED [(description, value, action, function, help, status-
class*)<exp>]+, [DEFAULT_VALUE, INITIAL_VALUE, size]
```

The attributes of the BIT_ENUMERATED type are specified in Table 170.

Table 170 – BIT_ENUMERATED attributes

Usage	Attribute	Description
m	description	See Table 168.
m	value	Unsigned integer representing the significance of the bit. If the value of the VARIABLE supplies at the specified bit positions a logical one, the associated description is displayed. Equal values are not allowed.
o	action	Reference to a METHOD that shall be executed when the bit is set.
o	function	Functional class of the bit. The functional class of a bit is the same as the CLASS of a VARIABLE. If no function is specified, the value of the function class defaults to the class of the variable. Therefore, if all the bits have the same function, then the CLASS of the VARIABLE shall be specified.
o	help	See Table 168.
o	size	See Table 168.
o	status-class	Meaning of the bit. A status bit may belong to more than one status-class (see Table 171). If the VARIABLE is not a status octet, then the bits do not have status classes.
o	DEFAULT_VALUE	Specifies the default setting for the VARIABLE. The DEFAULT_VALUE is available for all EDDL types. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as an INITIAL_VALUE. The DEFAULT_VALUE can also be built by an expression (see 7.40). Lexical structure DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	INITIAL_VALUE	When a device is instantiated for the first time, the INITIAL_VALUE of a VARIABLE is displayed. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned. Lexical structure INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)

Table 171 shows the status-classes for a VARIABLE of type BIT_ENUMERATED.

Table 171 – status–class attributes

Usage	Attribute	Description
m	DATA	An invalid data configuration.
m	HARDWARE	A hardware failure.
m	MISC	A miscellaneous condition.
m	MODE	The device is in a particular mode.
m	PROCESS	A problem with the process connected to the device.
m	SOFTWARE	A software failure.
m	CORRECTABLE	The bit can be cleared by the EDD application or the connected process.
m	SELF_CORRECTING	The bit will be reset without further intervention.
m	UNCORRECTABLE	The bit is neither self-correcting nor correctable.
m	EVENT	A one-time event.
m	STATE	The device is in a particular state.
m	COMM_ERROR	A communications failure in the other device.
o	IGNORE_IN_HANDHELD	A bit should be ignored in hand-held communicators.
o	IGNORE_IN_TEMPORARY_MASTER	A bit should be ignored in temporary master devices.
m	MORE	There is additional status information available from the device.
o	ALL	Impact all the outputs (see Table 172).
o	AO	Impacts one of the analog outputs (see Table 172).
m	BAD	The output is unreliable and should not be used for control (see Table 172).
o	DV	Impacts one of the dynamic variables (see Table 172).
o	TV	Impacts one of the transmitter variables (see Table 172).
m	DETAIL	The bit is summarized elsewhere in a bit of class summary.
m	SUMMARY	The bit is a logical combination of other bits of class detail.
o	INFO	The bit provides information about the status of the process or device.
o	WARNING	The bit provides a warning about the status of the process or device.
o	ERROR	The bit provides an error about the status of the process or device.
o	IGNORE_IN_HOST	The bit shall be ignored in a fixed, permanently attached host.

The status-classes ALL, AO, DV and TV provide more status information about the output of a device.

Lexical structure — ALL

ALL [AUTO_BAD, AUTO_GOOD, MANUAL_BAD, MANUAL_GOOD]

Lexical structure — AO

AO [AUTO_BAD, AUTO_GOOD, integer, MANUAL_BAD, MANUAL_GOOD]

Lexical structure — DV

DV [AUTO_BAD, AUTO_GOOD, integer, MANUAL_BAD, MANUAL_GOOD]

Lexical structure — TV

TV [AUTO_BAD, AUTO_GOOD, integer, MANUAL_BAD, MANUAL_GOOD]

The attributes are specified in Table 172.

Table 172 – ALL, AO, DV, TV attributes

Usage	Attribute	Description
s	AUTO_BAD	The value is coming from an application process and is invalid.
s	AUTO_GOOD	The value is coming from an application process and is invalid.
m	integer	The VARIABLE value coming from the device.
s	MANUAL_BAD	The value is not coming from an application process and is invalid.
s	MANUAL_GOOD	The value is not coming from an application process and is invalid.

7.33.2.2.2.4 ENUMERATED**Purpose**

A VARIABLE of type ENUMERATED is an unsigned integer defined as a set of predefined enumerated values.

Lexical structure

```
ENUMERATED [(description, value, help)<exp>]+, [DEFAULT_VALUE,
INITIAL_VALUE, size]
```

The attributes of the ENUMERATED type are specified in Table 173.

Table 173 – Enumerated types attributes

Usage	Attribute	Description
m	description	See Table 168.
m	value	Constant that specifies the VARIABLE value. The type is an unsigned integer. The enumeration list is optional but if defined a value and a description are required. Equal values are not allowed.
o	DEFAULT_VALUE	Specifies the default setting for the VARIABLE. The DEFAULT_VALUE is available for all EDDL types. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as an INITIAL_VALUE. The DEFAULT_VALUE can also be built by an expression, (see 7.40). Lexical structure DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	help	See Table 168.
o	INITIAL_VALUE	When a device is instantiated for the first time, the INITIAL_VALUE of a VARIABLE is displayed. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned. Lexical structure INITIAL_VALUE (unsigned_integer)
o	size	See Table 168.

7.33.2.2.2.5 INDEX**Purpose**

A VARIABLE of TYPE INDEX is an unsigned integer which is used to reference the elements of a REFERENCE_ARRAY (see 7.27). The integer attribute of ELEMENTS (see 7.27.2) specifies the allowable values of the VARIABLE.

When an INDEX VARIABLE is presented to the user, the text associated with the indices of the array shall be displayed, not the numeric value of the VARIABLE.

Lexical structure

INDEX (reference, size)<exp>

The attributes of the index type are specified in Table 174.

Table 174 – Index type attributes

Usage	Attribute	Description
o	size	Size of the VARIABLE in octets. This value is an integer constant greater than zero and has no upper bound. The default is 1. The REFERENCE_ARRAY shall contain only elements, which does not exceeds the index size of the variable.
m	reference	Reference to a REFERENCE_ARRAY instance.

7.33.2.2.2.6 OBJECT_REFERENCE

Purpose

VARIABLEs of the type OBJECT_REFERENCE allow the access to items of other COMPONENT instances. OBJECT_REFERENCES may be used to navigate within an object hierarchy, for example for a modular device (see IEC 61804-4).

Lexical structure

OBJECT_REFERENCE

7.33.2.2.2.7 ASCII, BITSTRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE

Purpose

String variable types include the following:

- ASCII – is for specifying a sequence of characters from the ISO Latin-1 character set.
- BITSTRING – is for specifying a sequence of bits.
- EUC (Extended Unicode) – is the internal code for specific characters (for example, China: EUC-CN, Taiwan EUC-TW). EUC is used to handle East Asian languages (ISO/IEC 10646-1).
- PACKED_ASCII – is a subset of ASCII produced by removing the two most significant bits of each ASCII character (see D.6.2.7).
- PASSWORD – is a string type and is intended for specifying password strings. Except for how the variable is presented to the user, password and ASCII string types are identical.
- VISIBLE – this string is an ordered sequence of characters from ISO/IEC 10646-1 and ISO 2375 character set.
- OCTET – is for specifying a sequence of unformatted binary data whose definition is determined by the implementation of the device.

Lexical structure — EUC

EUC [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Lexical structure — ASCII

ASCII [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Lexical structure – PACKED_ASCII

PACKED_ASCII [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Lexical structure — PASSWORD

PASSWORD [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Lexical structure — BITSTRING

BITSTRING [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Lexical structure — VISIBLE

VISIBLE [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Lexical structure — OCTET

OCTET [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

The attributes of the string types are specified in Table 175.

Table 175 – String types attributes

Usage	Attribute	Description
m	size	Size of the data type in octets or for BITSTRING number of bits. Size is an integer constant greater than zero and has no upper bound. This value is not optional; a size shall be specified.
o	DEFAULT_VALUE	Specifies the default setting for the VARIABLE. The DEFAULT_VALUE is available for all EDDL types. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as a INITIAL_VALUE. Lexical structure DEFAULT_VALUE (string)<exp>
o	INITIAL_VALUE	When a device is instantiated for the first time, the INITIAL_VALUE of a VARIABLE is displayed. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned. Lexical structure INITIAL_VALUE (string)
o	string	String constant. The enumeration list is optional but if defined a string and a description are required. Equal strings are not allowed.
o	description	See Table 168.
o	help	See Table 168.

7.33.2.2.2.8 BOOLEAN

Purpose

A VARIABLE of type BOOLEAN is a one byte unsigned integer with the value TRUE or FALSE. The value 0 reflects FALSE, any other value TRUE.

Lexical structure

BOOLEAN

7.33.2.3 CONSTANT_UNIT

Purpose

The CONSTANT_UNIT attribute is used if a VARIABLE has a units code which never changes. The CONSTANT_UNIT is specified as a string that will be displayed along with the value. A VARIABLE without a constant unit either has no units associated with it or the units are not constant.

Lexical structure

CONSTANT_UNIT (string)<exp>

The attribute of CONSTANT_UNIT is specified in Table 176.

Table 176 – CONSTANT_UNIT attribute

Usage	Attribute	Description
m	String	Units code string to be displayed.

7.33.2.4 DEFAULT_VALUE

Purpose

The DEFAULT_VALUE attribute specifies the default setting for the VARIABLE. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as the INITIAL_VALUE.

Lexical structure

DEFAULT_VALUE (expression)<exp>

The type of the expression shall be consistent with the type of the VARIABLE. The DEFAULT_VALUE may also be specified via the TYPE attribute of the VARIABLE.

The attribute of DEFAULT_VALUE is specified in Table 177.

Table 177 – DEFAULT_VALUE attribute

Usage	Attribute	Description
m	expression	The default value depends on the type of the variable: - DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER: see Table 168; - DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE: see Table 169; - BIT_ENUMERATED: see Table 170; - ENUMERATED: see Table 173; - ASCII, BITSTRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE: see Table 175.

7.33.2.5 INITIAL_VALUE**Purpose**

The INITIAL_VALUE attribute specifies the value of a VARIABLE that is displayed when a device is instantiated for the first time. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case, the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned.

Lexical structure

INITIAL_VALUE expression

The type of the expression shall be consistent with the type of the VARIABLE. The INITIAL_VALUE may also be specified via the TYPE attribute of the VARIABLE.

The attribute of INITIAL_VALUE is specified in Table 178.

Table 178 – INITIAL_VALUE attribute

Usage	Attribute	Description
m	expression	The initial value depends on the type of the variable: - DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER: see Table 168; - DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE: see Table 169; - BIT_ENUMERATED: see Table 170; - ENUMERATED: see Table 173; - ASCII, BITSTRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE: see Table 175.

7.33.2.6 POST_EDIT_ACTIONS**Purpose**

The POST_EDIT_ACTIONS attribute specifies METHODS that shall be executed after the user has finished editing the VARIABLE. The specified METHODS shall be executed in the

order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

The POST_EDIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

POST_EDIT_ACTIONS [reference]+, DEFINITION

The attributes of POST_EDIT_ACTIONS are specified in Table 179.

Table 179 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS attributes

Usage	Attribute	Description
o	reference	Reference to a METHOD instance or a METHOD instance with arguments.
o	DEFINITION	Lexical element (see 7.36.4).

7.33.2.7 POST_READ_ACTIONS

Purpose

The POST_READ_ACTIONS attribute specifies METHODS that shall be executed after the VARIABLE was read from the device, i.e. after the read response has been received and shall operate on the received data. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

If one of the PRE_READ_ACTIONS METHODS aborts, the VARIABLE shall not be read from the device and the POST_READ_ACTIONS METHODS shall not be invoked.

The POST_READ_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

POST_READ_ACTIONS [reference]+, DEFINITION

The attributes of POST_READ_ACTIONS are specified in Table 179.

7.33.2.8 POST_WRITE_ACTIONS

Purpose

The POST_WRITE_ACTIONS attribute specifies METHODS that shall be executed after the VARIABLE has been written to the device, i.e. after the write response has been received and shall operate on the received data, if there are any. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

If one of the PRE_WRITE_ACTIONS METHODS aborts, the VARIABLE shall not be written to the device and the POST_WRITE_ACTIONS METHODS shall not be invoked.

The POST_WRITE_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

POST_WRITE_ACTIONS [reference]+, DEFINITION

The attributes of POST_WRITE_ACTIONS are specified in Table 179.

7.33.2.9 PRE_EDIT_ACTIONS

Purpose

The PRE_EDIT_ACTIONS attribute specifies METHODS that shall be executed immediately when the VARIABLE is going to be edited. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

The PRE_EDIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

PRE_EDIT_ACTIONS [reference]+, DEFINITION

The attributes of PRE_EDIT_ACTIONS are specified in Table 179.

7.33.2.10 PRE_READ_ACTIONS

Purpose

The PRE_READ_ACTIONS attribute specifies METHODS that shall be executed before the VARIABLE is read. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

The PRE_READ_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

PRE_READ_ACTIONS [reference]+, DEFINITION

The attributes of PRE_READ_ACTIONS are specified in Table 179.

7.33.2.11 PRE_WRITE_ACTIONS

Purpose

The PRE_WRITE_ACTIONS attribute specifies METHODS that shall be executed before the VARIABLE is written to the device. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

The PRE_WRITE_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

PRE_WRITE_ACTIONS [reference]+, DEFINITION

The attributes of PRE_WRITE_ACTIONS are specified in Table 179.

7.33.2.12 READ_TIMEOUT, WRITE_TIMEOUT (void)

7.33.2.13 REFRESH_ACTIONS

Purpose

The REFRESH_ACTIONS attribute specifies METHODS that shall be executed whenever the value of the VARIABLE is requested. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

The REFRESH_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

REFRESH_ACTIONS [reference]+, DEFINITION

The attributes of REFRESH_ACTIONS are specified in Table 179.

7.33.2.14 STYLE (void)

The STYLE attribute is deprecated. HEIGHT and WIDTH are the replacement (see 7.36.7 and 7.36.17).

7.33.2.15 POST_RQSTUPDATE_ACTIONS

Purpose

POST_RQSTUPDATE_ACTIONS shall only be supported by device simulators.

The POST_RQSTUPDATE_ACTIONS attribute specifies METHODS that shall be executed when the VARIABLE's value is written to the device simulator. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

Lexical structure

POST_RQSTCHANGE_ACTIONS [(reference)<exp>]+

The attribute of POST_RQSTCHANGE_ACTIONS is specified in Table 180.

Table 180 – POST_USERCHANGE_ACTIONS, POST_RQSTUPDATE_ACTIONS attributes

Usage	Attribute	Description
m	reference	Reference to a METHOD.

7.33.2.16 POST_USERCHANGE_ACTIONS

Purpose

POST_USERCHANGE_ACTIONS shall only be supported by device simulators.

The POST_USERCHANGE_ACTIONS attribute specifies METHODS that shall be executed when the editing of the VARIABLE in the device simulation application is complete. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

Lexical structure

POST_USERCHANGE_ACTIONS [(reference) <exp>] +

The attribute of POST_USERCHANGE_ACTIONS is specified in Table 180.

7.34 VARIABLE_LIST

Purpose

A VARIABLE_LIST is a group of EDD communication objects (VARIABLE, VALUE_ARRAY or RECORDS). VARIABLE_LISTs are used to group objects for application convenience.

A VARIABLE_LIST may be transferred as one communication object.

Lexical structure

VARIABLE_LIST identifier, [MEMBERS, HELP, LABEL, RESPONSE_CODES]

The attributes of VARIABLE_LIST are specified in Table 181.

Table 181 – VARIABLE_LIST attributes

Usage	Attribute	Description
m	MEMBERS	Lexical element (see 7.36.12).
o	HELP	Lexical element (see 7.36.8).
o	LABEL	Lexical element (see 7.36.9).
o	RESPONSE_CODES	Lexical element (see 7.36.14).

7.35 WAVEFORM

7.35.1 General structure

Purpose

A WAVEFORM describes a data set that may be displayed by a GRAPH.

Lexical structure

WAVEFORM identifier [TYPE, EMPHASIS, EXIT_ACTIONS, HANDLING, HELP, INIT_ACTIONS, KEY_POINTS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY, VISIBILITY, Y_AXIS]

The attributes of WAVEFORM are specified in Table 182.

Table 182 – WAVEFORM attributes

Usage	Attribute	Description
m	TYPE	Lexical element (see 7.35.2.1).
o	EMPHASIS	Lexical element (see 7.36.5).
o	EXIT_ACTIONS	Lexical element (see 7.35.2.2).
o	HANDLING	Lexical element (see 7.36.6).
o	HELP	Lexical element (see 7.36.8).
o	INIT_ACTIONS	Lexical element (see 7.35.2.3).
o	KEY_POINTS	Lexical element (see 7.35.2.4).
o	LABEL	Lexical element (see 7.36.9).
o	LINE_COLOR	Lexical element (see 7.36.10).
o	LINE_TYPE	Lexical element (see 7.36.11).
o	REFRESH_ACTIONS	Lexical element (see 7.35.2.5).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).
o	Y_AXIS	Lexical element (see 7.35.2.6).

For each WAVEFORM not specifying a LINE_TYPE or LINE_COLOR, each line shall have a different appearance from the others on the GRAPH.

7.35.2 Specific attributes

7.35.2.1 TYPE

7.35.2.1.1 General structure

Purpose

The TYPE attribute specifies the type of data contained in the WAVEFORM.

Lexical structure

TYPE [XY, YT, HORIZONTAL, VERTICAL]

The attributes of TYPE are specified in Table 183.

Table 183 – TYPE attributes

Usage	Attribute	Description
s	XY	Lexical element (see 7.35.2.1.2.1).
s	YT	Lexical element (see 7.35.2.1.2.2).
s	HORIZONTAL	Lexical element (see 7.35.2.1.2.3).
s	VERTICAL	Lexical element (see 7.35.2.1.2.4).

7.35.2.1.2 Specific attributes

7.35.2.1.2.1 XY

Purpose

The XY attribute specifies a WAVEFORM that contains a list of (x,y) points.

Lexical structure

XY [X_VALUES, Y_VALUES, NUMBER_OF_POINTS]

The attributes of XY are specified in Table 184.

Table 184 – XY attributes

Usage	Attribute	Description
m	X_VALUES	Specifies the X coordinate of each point in the WAVEFORM. For each X coordinate specified in X_VALUES there shall be a corresponding Y coordinate specified in Y_VALUES. Lexical structure X_VALUES [reference, double, float, integer, unsigned_integer]+ The attributes of X_VALUES are specified in Table 190.
m	Y_VALUES	Specifies the Y coordinate of each point in the WAVEFORM. For each Y coordinate specified in Y_VALUES there shall be a corresponding X coordinate specified in X_VALUES. Lexical structure Y_VALUES [(reference, double, float, integer, unsigned_integer)+ The attributes of Y_VALUES are specified in Table 190.
o	NUMBER_OF_POINTS	Specifies the number of valid data points in X_VALUES and Y_VALUES. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES and Y_VALUES. Lexical structure NUMBER_OF_POINTS (integer)<exp>

7.35.2.1.2.2 YT

Purpose

The YT attribute specifies a WAVEFORM that contains a list of points that are defined via an initial X coordinate, an X increment between successive points and a list of Y values.

Lexical structure

YT [X_INITIAL, X_INCREMENT, Y_VALUES, NUMBER_OF_POINTS]

The attributes of YT are specified in Table 185.

Table 185 – YT attribute

Usage	Attribute	Description
m	X_INITIAL	Specifies the X coordinate of the first point in the WAVEFORM. Lexical structure X_INITIAL (reference)<exp> X_INITIAL (double)<exp> X_INITIAL (float)<exp> X_INITIAL (integer)<exp> X_INITIAL (unsigned_integer)<exp> The reference is a reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance or an element of a REFERENCE_ARRAY instance.
m	X_INCREMENT	Specifies the difference between the X coordinates of adjacent points in the WAVEFORM. Lexical structure X_INCREMENT (reference)<exp> X_INCREMENT (double)<exp> X_INCREMENT (float)<exp> X_INCREMENT (integer)<exp> X_INCREMENT (unsigned_integer)<exp> The reference is a reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance or an element of a REFERENCE_ARRAY instance.
m	Y_VALUES	Specifies the Y coordinate of each point in the WAVEFORM. Lexical structure Y_VALUES [(reference, double, float, integer, unsigned_integer)<exp>] + The attributes of Y_VALUES are specified in Table 190.
o	NUMBER_OF_POINTS	Specifies the number of valid data points in X_VALUES. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES. Lexical structure NUMBER_OF_POINTS (integer)<exp>

7.35.2.1.2.3 HORIZONTAL**Purpose**

The HORIZONTAL attribute specifies a WAVEFORM that contains horizontal lines.

Lexical structure

HORIZONTAL Y_VALUES

The attribute of HORIZONTAL is specified in Table 186.

Table 186 – HORIZONTAL attribute

Usage	Attribute	Description
m	Y_VALUES	Specifies the Y coordinate of each horizontal line in the WAVEFORM. Lexical structure Y_VALUES [reference, double, float, integer, unsigned_integer] + The attributes of Y_VALUES are specified in Table 190.

7.35.2.1.2.4 VERTICAL

Purpose

The VERTICAL attribute specifies a WAVEFORM that contains vertical lines.

Lexical structure

VERTICAL X_VALUES

The attribute of VERTICAL is specified in Table 187.

Table 187 – VERTICAL attribute

Usage	Attribute	Description
m	X_VALUES	Specifies the X coordinate of each vertical line in the WAVEFORM. Lexical structure X_VALUES [reference, double, float, integer, unsigned_integer]+ The attributes of X_VALUES are specified in Table 190.

7.35.2.2 EXIT_ACTIONS

Purpose

The EXIT_ACTIONS attribute specifies actions that are executed when the GRAPH containing the WAVEFORM is closed.

The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the remaining METHODS shall not be executed.

The EXIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

EXIT_ACTIONS [reference]+, DEFINITION

The attributes of EXIT_ACTIONS are specified in Table 188.

7.35.2.3 INIT_ACTIONS

Purpose

The INIT_ACTIONS attribute specifies actions that are executed before the WAVEFORM is initially displayed by a GRAPH. These methods are generally used to load the WAVEFORM with data from a device or a FILE.

The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the remaining METHODS shall not be executed. Furthermore, the EDD application shall not draw the WAVEFORM if a METHOD exits for an unplanned reason.

The INIT_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

INIT_ACTIONS [reference]+, DEFINITION

The attributes of INIT_ACTIONS are specified in Table 188.

Table 188 – EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS attributes

Usage	Attribute	Description
o	reference	Reference to a METHOD instance or a METHOD instance with arguments.
o	DEFINITION	Lexical element (see 7.36.4).

7.35.2.4 KEY_POINTS

7.35.2.4.1 General structure

Purpose

The KEY_POINTS attribute specifies key points in the WAVEFORM that should be highlighted by the EDD application. The key points need not directly correspond to the data points specified via the TYPE attribute. The way in which these points are highlighted is defined by the EDD specification. By default, a WAVEFORM without a KEY_POINTS attribute should not have any of its points highlighted.

Lexical structure

KEY_POINTS [X_VALUES, Y_VALUES]

The attributes of KEY_POINTS are specified in Table 189.

Table 189 – KEY_POINTS attributes

Usage	Attribute	Description
m	X_VALUES	X coordinate of each KEY_POINT.
m	Y_VALUES	Y coordinate of each KEY_POINT.

7.35.2.4.2 Specific attributes

7.35.2.4.2.1 X_VALUES

Purpose

The X_VALUES attribute specifies the X coordinate of each point that is to be highlighted. For each X coordinate specified in X_VALUES there shall be a corresponding Y coordinate specified in Y_VALUES.

Lexical structure

X_VALUES [reference, double, float, integer, unsigned_integer]+

The attributes of X_VALUES are specified in Table 190.

Table 190 – X_VALUES, Y_VALUES attributes

Usage	Attribute	Description
o	reference	Reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance, an element of a REFERENCE_ARRAY instance, a VALUE_ARRAY instance or a REFERENCE_ARRAY instance.
o	double	Double-precision floating-point constant.
o	float	Single-precision floating-point constant.
o	integer	Integer constant.
o	unsigned_integer	Unsigned integer constant.

7.35.2.4.2.2 Y_VALUES**Purpose**

The Y_VALUES attribute specifies the Y coordinate of each point that is to be highlighted. For each Y coordinate specified in Y_VALUES, there shall be a corresponding X coordinate specified in X_VALUES.

Lexical structure

Y_VALUES [reference, double, float, integer, unsigned_integer] +

The attributes of Y_VALUES are specified in Table 190.

7.35.2.5 REFRESH_ACTIONS**Purpose**

The REFRESH_ACTIONS attribute specifies actions that are executed when changes related to the X_AXIS or Y_AXIS are made or if a CYCLE_TIME is defined within the GRAPH each time the CYCLE_TIME elapses.

The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the remaining METHODS shall not be executed. Furthermore, the X_VALUES and Y_VALUES of the WAVEFORM shall be restored to their original values by the EDD application, and the WAVEFORM on the GRAPH shall not be redrawn.

The REFRESH_ACTIONS attribute shall not be expressed using a conditional IF, IF-ELSE, SELECT. If conditional behavior is required, the methods invoked by the action shall accommodate it.

Lexical structure

REFRESH_ACTIONS [reference] +, DEFINITION

The attributes of REFRESH_ACTIONS are specified in Table 188.

7.35.2.6 Y_AXIS**Purpose**

The Y_AXIS attribute specifies the way the Y-axis of the WAVEFORM is displayed. Each WAVEFORM on a GRAPH shares a common X_AXIS but each WAVEFORM on a GRAPH has its own Y_AXIS. By default, the display of the Y-axis of a GRAPH without a Y_AXIS attribute is inferred from the data in the WAVEFORM.

Lexical structure

Y_AXIS (reference) <exp>

The attribute of Y_AXIS is specified in Table 191.

Table 191 – Y_AXIS attribute

Usage	Attribute	Description
m	reference	Reference to an AXIS instance.

7.36 Common attributes

7.36.1 CLASSIFICATION

Purpose

The CLASSIFICATION attribute is a single enumerated value that indicates the (possibly multiple) levels classification for this device.

Lexical structure

CLASSIFICATION [see attributes in Table 192]

The attributes of CLASSIFICATION are specified in Table 192.

Table 192 – CLASSIFICATION attributes

Usage	Attribute	Description
s	ACTUATOR	The component represents this.
s	ACTUATOR_ELECTRO_PNEUMATIC	The component represents this.
s	ACTUATOR_ELECTRIC	The component represents this.
s	ACTUATOR_HYDRAULIC	The component represents this.
s	CONVERTER	The component represents this.
s	CONTROLLER	The component represents this.
s	DISCRETE_IN	The component represents this.
s	DISCRETE_OUT	The component represents this.
s	ELECTRICAL_DISTRIBUTION	The component represents this.
s	ELECTRICAL_DISTRIBUTION_POWER_MONITORING	The component represents this.
s	FREQUENCY_CONVERTER	The component represents this.
s	INDICATOR	The component represents this.
s	NETWORK	The component represents this.
s	NETWORK_COMMUNICATION_SERVICE_PROVIDER	The component represents this.
s	NETWORK_COMPONENT	The component represents this.
s	NETWORK_COMPONENT_WIRELESSADAPTER	The component represents this.
s	NETWORK_CONNECTION_POINT	The component represents this.
s	REMOTEIO	The component represents this.
s	RECORDER	The component represents this.
s	SENSOR	The component represents this.
s	SENSOR_ACOUSTIC	The component represents this.
s	SENSOR_ANALYTIC	The component represents this.
s	SENSOR_ANALYTIC_GAS	The component represents this.

Usage	Attribute	Description
s	SENSOR_ANALYTIC_LIQUID	The component represents this.
s	SENSOR_CONCENTRATION	The component represents this.
s	SENSOR_DENSITY	The component represents this.
s	SENSOR_DENSITY_RADIOMETRIC	The component represents this.
s	SENSOR_FLOW	The component represents this.
s	SENSOR_FLOW_CORIOLIS	The component represents this.
s	SENSOR_FLOW_ELECTRO_MAGNETIC	The component represents this.
s	SENSOR_FLOW_MECHANICAL	The component represents this.
s	SENSOR_FLOW_RADIOMETRIC	The component represents this.
s	SENSOR_FLOW_THERMAL	The component represents this.
s	SENSOR_FLOW_ULTRASONIC	The component represents this.
s	SENSOR_FLOW_VORTEX_COUNTER	The component represents this.
s	SENSOR_LEVEL	The component represents this.
s	SENSOR_LEVEL_BUOYANCY	The component represents this.
s	SENSOR_LEVEL_CAPACITIVE	The component represents this.
s	SENSOR_LEVEL_ECHO	The component represents this.
s	SENSOR_LEVEL_HYDROSTATIC	The component represents this.
s	SENSOR_LEVEL_RADIOMETRIC	The component represents this.
s	SENSOR_MULTIVARIABLE	The component represents this.
s	SENSOR_POSITION	The component represents this.
s	SENSOR_PRESSURE	The component represents this.
s	SENSOR_TEMPERATURE	The component represents this.
s	SEMICONDUCTOR	The component represents this.
s	SWITCHGEAR	The component represents this.
s	UNIVERSAL	The component represents this.
s	string	The component represents this.

7.36.2 COMPONENT_PARENT

Purpose

The COMPONENT_PARENT attribute is a reference to a COMPONENT_FOLDER. COMPONENT_PARENT can be used instead of PROTOCOL and CLASSIFICATION to simplify (shorten) the COMPONENT_PATH declaration. If the PROTOCOL and/or CLASSIFICATION are defined in the COMPONENT_PARENT (or its parent) the PROTOCOL and/or CLASSIFICATION shall not be redefined. If the COMPONENT_PARENT is specified, the COMPONENT_PATH shall be defined.

NOTE The COMPONENT_PARENT is predefined by the HCF and FF EDD library structure.

Lexical structure

COMPONENT_PARENT reference

The attribute of COMPONENT_PARENT is specified in Table 193.

Table 193 – COMPONENT_PARENT attribute

Usage	Attribute	Description
m	reference	Parent component in the catalog.

7.36.3 COMPONENT_PATH**Purpose**

The COMPONENT_PATH attribute allows the EDD manufacturer to designate the location in the catalog of an EDD application. The path only specifies the manufacturer specific part of the location without the PROTOCOL, CLASSIFICATION and MANUFACTURER. If a COMPONENT_PARENT is used the PROTOCOL, CLASSIFICATION and MANUFACTURER may not be defined and the COMPONENT_PATH defines only the sub hierarchy.

NOTE The COMPONENT_PATH is predefined by the HCF and FF EDD library structure.

Lexical structure

COMPONENT_PATH string

The attribute of COMPONENT_PATH is specified in Table 194.

Table 194 – COMPONENT_PATH attribute

Usage	Attribute	Description
m	string	Manufacturer specific path of the component in the catalog.

7.36.4 DEFINITION**Purpose**

The DEFINITION attribute specifies actions to be performed and is based on a procedural programming language (for example, programming language-C). In the DEFINITIONS, METHODS and Builtins (see IEC 61804-5) may be called. Within a DEFINITION EDD basic elements can be used.

The following items shall be supported.

- Basic data types (see D.6.1).
- Arrays.
- Operators (see A.3).
- Statements.
- Invocation of METHODS and Builtins.
- METHODS with arguments (call by value, call by reference):
 - Call by value is used to pass a copy of the data to the METHOD. Changes to the copy within the METHOD do not change the original data.
 - Call by reference is used to access data directly. Changes affect the original data.

Lexical structure

DEFINITION compound-statement

The attribute of DEFINITION is specified in Table 195.

Table 195 – DEFINITION attribute

Usage	Attribute	Description
o	compound-statement	Block of instructions for a supported procedural programming language.

7.36.5 EMPHASIS

Purpose

The EMPHASIS attribute specifies whether or not a user interface element should be emphasized when it is displayed. The way in which it is emphasized is defined by the EDD application. By default, a user interface element without an EMPHASIS attribute should not be emphasized.

Lexical structure

EMPHASIS (FALSE, TRUE) <exp>

The attributes of EMPHASIS are specified in Table 196.

Table 196 – EMPHASIS attributes

Usage	Attribute	Description
s	FALSE	The user interface element should not be emphasized.
s	TRUE	The user interface element should be emphasized.

7.36.6 HANDLING

Purpose

The HANDLING attribute specifies the operations that can be performed on an element. By default, an element without a HANDLING attribute can be read and written.

Lexical structure

HANDLING ([READ, WRITE] +) <exp>

The attributes of HANDLING are specified in Table 197.

Table 197 – HANDLING attributes

Usage	Attribute	Description
s	READ	The element may be read.
s	WRITE	The element may be written.

7.36.7 HEIGHT

Purpose

The HEIGHT attribute specifies the vertical size of a user interface element when it is displayed. A VARIABLE without a HEIGHT attribute defined is handled as if it has the HEIGHT XXX_SMALL. A GRAPH, a CHART, and a GRID without a HEIGHT attribute specified are handled as if they have the HEIGHT MEDIUM.

Lexical structure

HEIGHT (XXX_SMALL, XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE)

The attributes of HEIGHT are specified in Table 198.

Table 198 – HEIGHT/WIDTH attribute

Usage	Attribute	Description
s	XXX_SMALL	The user interface element should occupy an extra, extra extra small portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies one column. If this value is used for HEIGHT the interface element occupies the minimum height that is possible for a standard input field. XXX_SMALL is only allowed for VARIABLES.
s	XX_SMALL	The user interface element should occupy an extra, extra small portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies one column. If this value is used for HEIGHT the interface element occupies three times the minimum height that is possible for a standard input field.
s	X_SMALL	The user interface element should occupy an extra small portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies one column. If this value is used for HEIGHT the interface element occupies five times the minimum height that is possible for a standard input field.
s	SMALL	The user interface element should occupy a small portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies two columns. If this value is used for HEIGHT the interface element occupies seven times the minimum height that is possible for a standard input field.
s	MEDIUM	The user interface element should occupy a medium portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies three columns. If this value is used for HEIGHT the interface element occupies eight times the minimum height that is possible for a standard input field.
s	LARGE	The user interface element should occupy a large portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies four columns. If this value is used for HEIGHT the interface element occupies nine times the minimum height that is possible for a standard input field.
s	X_LARGE	The user interface element should occupy an extra large portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies five columns. If this value is used for HEIGHT the interface element occupies eleven times the minimum height that is possible for a standard input field.
s	XX_LARGE	The user interface element should occupy an extra, extra large portion of the window in which it is being displayed. If this value is used for WIDTH the interface element occupies five columns. If this value is used for HEIGHT the interface element occupies thirteen times the minimum height that is possible for a standard input field.

7.36.8 HELP

Purpose

The HELP attribute specifies text, which provides a description of the construct.

NOTE This text is intended to be used by EDD applications for user information.

Lexical structure

HELP (string)<exp>

The attribute of HELP is specified in Table 199.

Table 199 – HELP attribute

Usage	Attribute	Description
m	string	User readable text.

7.36.9 LABEL**Purpose**

The LABEL attribute specifies the displayed designation of an element.

Lexical structure

`LABEL (string)<exp>`

The attribute of LABEL is specified in Table 200.

Table 200 – LABEL attribute

Usage	Attribute	Description
m	string	User readable text.

7.36.10 LINE_COLOR**Purpose**

The LINE_COLOR attribute specifies optionally the color that should be used to display a user interface element. A user interface element without a LINE_COLOR attribute will be displayed using a default color defined by the EDD application.

Lexical structure

`LINE_COLOR (integer, reference)<exp>`

Colors are specified via an RGB value, as defined in W3C Cascading Style Sheets, Level 2.

The attributes of LINE_COLOR are specified in Table 201.

Table 201 – LINE_COLOR attributes

Usage	Attribute	Description
s	integer	Color to be used to display the user interface element.
s	reference	Reference to a VARIABLE instance containing the color to be used to display the user interface element.

7.36.11 LINE_TYPE**Purpose**

The LINE_TYPE attribute is used to differentiate types of data by line color, line pattern, line thickness, etc. Within a user interface element the contained items with the same LINE_TYPE attribute, except for DATA, should be displayed in the same fashion.

By default, an item without a LINE_TYPE attribute uses the DATA line type. For line type DATA (no number) the EDD application shall display the items differently.

Lexical structure

LINE_TYPE ([DATA, n]*, LOW_LOW_LIMIT, LOW_LIMIT, HIGH_LIMIT, HIGH_HIGH_LIMIT, TRANSPARENT) <exp>

The attributes of LINE_TYPE are specified in Table 202.

Table 202 – LINE_TYPE attribute

Usage	Attribute	Description
s	DATA	The user interface element contains data.
c	n	Integer between 1 and 9, which identifies distinct DATA line types if more than one is present.
s	LOW_LOW_LIMIT	The user interface element contains a low low limit.
s	LOW_LIMIT	The user interface element contains a low limit.
s	HIGH_LIMIT	The user interface element contains a high limit.
s	HIGH_HIGH_LIMIT	The user interface element contains a high high limit.
s	TRANSPARENT	Only the user interface element data points are visible.

7.36.12 MEMBERS

Purpose

The MEMBERS attribute defines members of a basic construct. Each member specifies one item in the group, and is defined by a set of four attributes (identifier, reference, description, help).

The MEMBERS attribute of a FILE shall not be expressed using a conditional IF, IF-ELSE, SELECT.

Lexical structure

MEMBERS [(identifier, reference, description, help) <exp>] +

The attributes of MEMBERS are specified in Table 203.

Table 203 – MEMBERS attributes

Usage	Attribute	Description
m	identifier	Name by which the item may be referenced. A specific identifier name can be reused in the MEMBERS attribute of multiple items as long as the items are of the same type.
m	reference	Reference to an EDD item. The type of the reference depends on the basic constructs: ▪ CHART: reference to a SOURCE instance; ▪ COLLECTION: reference to a BLOCK_A, BLOCK_B, CHART, COLLECTION, EDIT_DISPLAY, FILE, GRAPH, LIST, MENU, METHOD, PLUGIN, RECORD, REFERENCE_ARRAY, VALUE_ARRAY, VARIABLE or VARIABLE_LIST instance; ▪ FILE or LIST: reference to a COLLECTION, LIST, RECORD, REFERENCE_ARRAY, VALUE_ARRAY or VARIABLE instance. For a COLLECTION, the type of the COLLECTION shall be LIST, RECORD, REFERENCE_ARRAY, VALUE_ARRAY, or VARIABLE; ▪ GRAPH: reference to a WAVEFORM instance; ▪ RECORD: reference to a VARIABLE instance; ▪ SOURCE: reference is a reference to a VARIABLE instance. ▪ VARIABLE_LIST: reference to a RECORD, or VALUE_ARRAY or VARIABLE instance.
o	description	Short description of the item.
o	help	Help text for the item.

7.36.13 PROTOCOL

Purpose

The PROTOCOL attribute indicates the communication protocol used to communicate to the device modeled by this EDD. The PROTOCOL is used to separate the EDDs within the catalog.

Lexical structure

PROTOCOL [PROFIBUS_DP, PROFIBUS_PA, PROFINET, FF, HART, string]

The attributes of PROTOCOL are specified in Table 204.

Table 204 – PROTOCOL attributes

Usage	Attribute	Description
s	FF	The component is a Fieldbus Foundation component.
s	HART	The component is a HART component.
s	PROFIBUS_DP	The component is a PROFIBUS DP component.
s	PROFIBUS_PA	The component is a PROFIBUS PA component.
s	PROFINET	The component is a PROFINET component.
s	string	Specifies any protocol. The protocol shall be supported by the used EDD application.

7.36.14 RESPONSE_CODES

Purpose

The RESPONSE_CODES attribute specifies values a device may return as error information. Each VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST or COMMAND can have its own set of RESPONSE_CODES.

Two lexical structures for the RESPONSE_CODES attribute are provided:

- the referenced RESPONSE_CODES contains a reference to a RESPONSE_CODES instance;
- the embedded RESPONSE_CODES allows to specify the RESPONSE_CODES elements directly at the element instance.

Lexical structure for referenced RESPONSE_CODES

RESPONSE_CODES (reference)<exp>

The attribute of the referenced RESPONSE_CODES is specified in Table 205.

Table 205 – RESPONSE_CODES attribute

Usage	Attribute	Description
m	reference	Reference to a RESPONSE_CODES instance.

Lexical structure for embedded RESPONSE_CODES

RESPONSE_CODES [(description, integer, [DATA_ENTRY_ERROR, DATA_ENTRY_WARNING, MISC_ERROR, MISC_WARNING, MODE_ERROR, PROCESS_ERROR, SUCCESS], help)<exp>]+

The attributes of the embedded RESPONSE_CODES are specified in Table 157.

7.36.15 SUPPLIED_INTERFACE

Purpose

The SUPPLIED_INTERFACE attribute lists all interfaces this component supports.

Lexical structure

SUPPLIED_INTERFACE [interface-reference]+

The attribute of SUPPLIED_INTERFACE is specified in Table 206.

Table 206 – SUPPLIED_INTERFACE attribute

Usage	Attribute	Description
m	interface-reference	List of interface identifiers.

7.36.16 VALIDITY

Purpose

The VALIDITY attribute specifies whether an element is valid or invalid. An element without a VALIDITY is always valid.

VALIDITY should be expressed using a conditional IF, IF-ELSE, SELECT. If a user interface element is invalid, it should not be displayed. A METHOD shall abort when an invalid VARIABLE is accessed.

Lexical structure

VALIDITY [(FALSE, TRUE)<exp>]

The attributes of VALIDITY are specified in Table 207.

Table 207 – VALIDITY attributes

Usage	Attribute	Description
s	FALSE	The element is invalid.
s	TRUE	The element is valid.

7.36.17 WIDTH

Purpose

The WIDTH attribute specifies the horizontal size of a user interface element when it is displayed. A VARIABLE without a WIDTH attribute specified is handled as if it has the WIDTH XXX_SMALL. A GRAPH, a CHART, and a GRID without a WIDTH attribute specified are handled as if they have the WIDTH MEDIUM.

Lexical structure

WIDTH (XXX_SMALL, XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE)

The attributes of WIDTH are specified in Table 198.

7.36.18 PRIVATE

Purpose

The PRIVATE attribute specifies whether an element is exposed outside the EDD application. If the PRIVATE attribute is not specified, the element is always public.

Lexical structure

PRIVATE [(FALSE, TRUE)]

The attributes of PRIVATE are specified in Table 208.

Table 208 – PRIVATE attributes

Usage	Attribute	Description
s	FALSE	The element is public. The default value of PRIVATE is FALSE.
s	TRUE	The element is private.

7.36.19 VISIBILITY

Purpose

The VISIBILITY attribute specifies whether an element is visible or not. An element without a VISIBILITY is always visible.

VISIBILITY should be expressed using a conditional IF, IF-ELSE, SELECT. VISIBILITY does not affect the screen layout. In the screen layout, the same space is reserved as if the element was shown.

Lexical structure

VISIBILITY [(FALSE, TRUE)<exp>]

The attributes of VISIBILITY are specified in Table 209.

Table 209 – VISIBILITY attributes

Usage	Attribute	Description
s	FALSE	The element is not visible.
s	TRUE	The element is visible. The default value of VISIBILITY is TRUE.

7.36.20 WRITE_MODE

Purpose

The WRITE_MODE attribute specifies the minimum target mode requirements that shall be set before the value can be transferred to the device.

Lexical structure

WRITE_MODE [(MODE_OUT_OF_SERVICE, MODE_INITIALIZATION, MODE_LOCAL_OVERRIDE, MODE_MANUAL, MODE_AUTOMATIC, MODE_CASCADE, MODE_REMOTE_CASCADE, MODE_REMOTE_OUTPUT)<exp>]

The attributes of WRITE_MODE are specified in Table 210.

Table 210 – WRITE_MODE attributes

Usage	Attribute	Priority	Description
s	MODE_OUT_OF_SERVICE	7 – highest	Out of Service (O/S)
s	MODE_INITIALIZATION	6	Initialization Manual (IMan)
s	MODE_LOCAL_OVERRIDE	5	Local Override (LO)
s	MODE_MANUAL	4	Manual (Man)
s	MODE_AUTOMATIC	3	Automatic (Auto)
s	MODE_CASCADE	2	Cascade (Cas)
s	MODE_REMOTE_CASCADE	1	Remote-Cascade (RCas)
s	MODE_REMOTE_OUTPUT	0 – lowest	Remote-Output (ROut)

7.36.21 IDENTITY

Purpose

The IDENTITY attribute specifies a filename. Specifying the IDENTITY allows access to files that may be created or used by applications external to the EDD application.

Lexical structure

IDENTITY file-name

The attribute of IDENTITY is specified in Table 211.

Table 211 – IDENTITY attribute

Usage	Attribute	Description
m	file-name	String literal that identifies the filename.

7.37 Conditional expression

Purpose

Conditional expressions are used to declare alternative values or to calculate values for an attribute of a basic element. The value of the expression is evaluated during the execution time. The IF or SELECT conditional can be used to enable or disable the referenced instances of a list of ITEMS attributes of MENU, etc.

Three kinds of conditional expressions may be used.

- An IF conditional is used for specifying an attribute that has two alternative definitions. The expression specified in the IF conditional is evaluated. If the result is non-zero, the attribute is specified by a then-clause; otherwise, it is specified by an else-clause.
- A SELECT conditional is used for specifying an attribute that has several alternative definitions. The select-expression is evaluated and compared with every integer_n of CASE. If a match is found, the appropriate select-clause is valid. If no matching integer_n is found, the default-clause following DEFAULT is used.
- A primary, unary or binary expression (see 7.39.7). This kind of conditional expression can only be used for numeric values (e.g. DEFAULT_VALUE, MIN_VALUE, SLOT, INDEX).

EXAMPLE It is possible to define more than one value for a DEFAULT_VALUE, MIN_VALUE, etc. using an IF or SELECT conditional.

Lexical structure for the IF conditional

IF (if-expression), (then-clause)+ ELSE (else-clause)+

The attributes are specified in Table 212.

Lexical structure for the SELECT conditional

SELECT select-expression, (CASE integer_n, select-clause)+, (DEFAULT, default-clause)

The attributes are specified in Table 212.

Table 212 – IF, SELECT conditional

Usage	Attribute	Description
m	if-expression	Evaluated to determine whether the then-clause or the else-clause is used to define an attribute. If the if-expression contains a reference to a VARIABLE instance, the value of the referenced VARIABLE is used.
m	then-clause	The definition for the attribute if the value of condition is non-zero. The clause structure depends on the attribute being defined. It can also take the form of another conditional.
o	else-clause	The definition for the attribute if the value of condition is zero. The clause structure depends on the attribute being defined. It can also take the form of another conditional.
m	select-expression	Evaluated to determine which select-clause is used to define an attribute. If the select-expression contains a reference to a VARIABLE instance, the value of the referenced VARIABLE is used.
m	integer_n	A number which identifies the according CASE. If integer matches with integer_n, the according select_clause is used.
m	select-clause	The definition for the attribute for each CASE, and the DEFAULT definition. The structure of each clause depends on the attribute being defined. Regardless of the attribute, each clause can also take the form of another conditional.
o	default-clause	The definition for the DEFAULT definition. The structure of default-clause depends on the attribute being defined. Regardless of the attribute, each clause can also take the form of another conditional.

If the

- then-clause,
- else-clause,
- select-clause,
- default-clause

contain references to METHOD instances or Builtins, the returned value of the referenced METHOD or Builtin shall be used within the clause. The return values shall be type-compatible.

7.38 Referencing

7.38.1 Referencing an EDD instance

Purpose

References are used within an EDD to refer to other EDD items.

EXAMPLE A PRE_EDIT_ACTIONS of VARIABLE refers to METHOD instances defined elsewhere in the EDD.

Lexical structure

The attribute is specified in Table 213.

Table 213 – Referencing an EDD instance

Usage	Attribute	Description
m	identifier	Identifier of an EDD instance.

7.38.2 Referencing bits of a BIT_ENUMERATED VARIABLE

Purpose

This referencing is used to access a bit of a BIT_ENUMERATED VARIABLE instance.

Lexical structure

variable-reference, bit-mask

The attributes are specified in Table 214.

Table 214 – Referencing elements of VARIABLE

Usage	Attribute	Description
m	variable-reference	Reference to a VARIABLE instance. The type of the VARIABLE instance shall be BIT_ENUMERATED.
m	bit-mask	Bit-mask which identifies a single bit within the variable. The bit-mask shall be a constant expression.

7.38.3 Referencing members of a RECORD

Purpose

This referencing is used to access a member of a RECORD instance.

Lexical structure

record-reference, member-identifier

The attributes are specified in Table 215.

Table 215 – Referencing elements of RECORD

Usage	Attribute	Description
m	record-reference	Reference to a RECORD instance.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.4 Referencing elements of a VALUE_ARRAY

Purpose

This referencing is used to access an element of a VALUE_ARRAY instance.

Lexical structure

value-array-reference, expression

The attributes are specified in Table 183.

Table 216 – Referencing elements of VALUE_ARRAY

Usage	Attribute	Description
m	value-array-reference	Reference to a VALUE_ARRAY instance.
m	expression	Expression which shall be evaluated to a non-negative integer that is within the index range of the VALUE_ARRAY; for a description of expressions, see 7.40.

7.38.5 Referencing members of a COLLECTION**Purpose**

This referencing is used to access a member of a COLLECTION instance.

Lexical structure

collection-reference, member-identifier

The attributes are specified in Table 217.

Table 217 – Referencing members of COLLECTION

Usage	Attribute	Description
m	collection-reference	Reference to a COLLECTION instance.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.6 Referencing elements of a REFERENCE_ARRAY**Purpose**

This referencing is used to access an element of a REFERENCE_ARRAY instance.

Lexical structure

reference-array-reference, expression

The attributes are specified in Table 218.

Table 218 – Referencing members of REFERENCE_ARRAY

Usage	Attribute	Description
m	reference-array-reference	Reference to a REFERENCE_ARRAY instance.
m	expression	Expression, which shall be evaluated to a non-negative integer that is within the index range of the REFERENCE_ARRAY; for a description of expressions, see 7.40.

7.38.7 Referencing members of a VARIABLE_LISTS**Purpose**

This referencing is used to access a member of a VARIABLE_LISTS instance.

Lexical structure

variable-list-reference, member-identifier

The attributes are specified in Table 219.

Table 219 – Referencing members of VARIABLE_LISTS

Usage	Attribute	Description
m	variable-list-reference	Reference to a VARIABLE_LISTS instance.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.8 Referencing elements of BLOCK_A PARAMETERS

Purpose

This referencing is used to access an element of the PARAMETERS attribute of a BLOCK_A instance.

Lexical structure

PARAM, member-identifier

The attribute is specified in Table 220.

Table 220 – Referencing members of a BLOCK_A PARAMETERS

Usage	Attribute	Description
m	member-identifier	Identifier of an item of MEMBERS.

7.38.9 Referencing elements of BLOCK_A PARAMETER_LISTS

Purpose

This referencing is used to access an element of the PARAMETER_LISTS attribute of a BLOCK_A instance.

Lexical structure

PARAM_LIST, member-identifier

The attribute is specified in Table 221.

Table 221 – Referencing members of BLOCK_A PARAMETER_LISTS

Usage	Attribute	Description
m	member-identifier	Identifier of an item of MEMBERS.

7.38.10 Referencing elements of BLOCK_A LOCAL_PARAMETERS

Purpose

This referencing is used to access an element of the LOCAL_PARAMETERS attribute of a BLOCK_A instance.

Lexical structure

LOCAL_PARAM, member-identifier

The attribute is specified in Table 222.

Table 222 – Referencing members of BLOCK_A LOCAL_PARAMETER

Usage	Attribute	Description
m	member-identifier	Identifier of an item of MEMBERS.

7.38.11 Referencing BLOCK_A CHARACTERISTICS**Purpose**

This referencing is used to access the CHARACTERISTICS of a BLOCK_A instance.

Lexical structure

BLOCK, characteristics-identifier

The attribute is specified in Table 223.

Table 223 – Referencing BLOCK_A CHARACTERISTICS

Usage	Attribute	Description
m	characteristics-identifier	Identifier of the CHARACTERISTICS reference.

7.38.12 Referencing members of a FILE**Purpose**

This referencing is used to access a member of a FILE instance.

Lexical structure

file-reference, member-identifier

The attributes are specified in Table 224.

Table 224 – Referencing members of FILE

Usage	Attribute	Description
m	file-reference	Reference to a FILE instance.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.13 Referencing elements of a LIST**Purpose**

This referencing is used to access an element of a LIST instance.

Lexical structure

list-reference, (expression, FIRST, LAST)

The attributes are specified in Table 225.

Table 225 – Referencing elements of LIST

Usage	Attribute	Description
m	list-reference	Reference to a LIST instance.
s	expression	Expression which shall be evaluated to a non-negative integer that is within the index range of the LIST (for a description of expressions, see 7.39.7).
s	FIRST	First element of the LIST.
s	LAST	Last element of the LIST.

7.38.14 Referencing members of a CHART**Purpose**

This referencing is used to access a member of a CHART instance.

Lexical structure

chart-reference, member-identifier

The attributes are specified in Table 226.

Table 226 – Referencing members of CHART

Usage	Attribute	Description
m	chart-reference	Reference to a CHART instance.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.15 Referencing members of a GRAPH**Purpose**

This referencing is used to access a member of a GRAPH instance.

Lexical structure

graph-reference, member-identifier

The attributes are specified in Table 227.

Table 227 – Referencing members of GRAPH

Usage	Attribute	Description
m	graph-reference	Reference to a GRAPH instance.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.16 Referencing members of a SOURCE**Purpose**

This referencing is used to access a member of a GRAPH instance.

Lexical structure

source-reference, member-identifier

The attributes are specified in Table 228.

Table 228 – Referencing members of SOURCE

Usage	Attribute	Description
m	source-reference	Reference to a SOURCE instance.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.17 Referencing AXIS of a GRAPH, SOURCE, WAVEFORM**Purpose**

This referencing is used to access an AXIS of a GRAPH, SOURCE or WAVEFORM instance.

Lexical structure

reference, (X_AXIS, Y_AXIS)

The attributes are specified in Table 229.

Table 229 – Referencing AXIS of a GRAPH, SOURCE, WAVEFORM

Usage	Attribute	Description
m	reference	Reference to a GRAPH instance, a reference to a SOURCE instance, or a reference to a WAVEFORM instance.
c/o	X_AXIS	Shall only be used with GRAPH; Y_AXIS used in all other cases.
c/o	Y_AXIS	Shall only be used with SOURCE and WAVEFORM; X_AXIS used in all other cases.

7.38.18 Referencing PARAMETERS of specific BLOCK_A instance**Purpose**

This referencing is used to access a member of the PARAMETERS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, PARAM, member-identifier

The attributes are specified in Table 230.

Table 230 – Referencing PARAMETERS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.19 Referencing LOCAL_PARAMETERS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the LOCAL_PARAMETERS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, LOCAL_PARAM, member-identifier

The attributes are specified in Table 231.

Table 231 – Referencing LOCAL_PARAMETERS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.20 Referencing CHARACTERISTICS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the CHARACTERISTICS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, BLOCK, member-identifier

The attributes are specified in Table 232.

Table 232 – Referencing CHARACTERISTICS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.21 Referencing CHARTS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the CHARTS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, CHART, member-identifier

The attributes are specified in Table 233.

Table 233 – Referencing CHARTS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.22 Referencing LISTS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the LISTS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, LIST, member-identifier

The attributes are specified in Table 234.

Table 234 – Referencing LISTS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.23 Referencing GRAPHS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the GRAPHS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, GRAPH, member-identifier

The attributes are specified in Table 235.

Table 235 – Referencing GRAPHS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.24 Referencing GRIDS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the GRIDS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, GRID, member-identifier

The attributes are specified in Table 236.

Table 236 – Referencing GRIDS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.25 Referencing MENUS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the MENUS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, MENU, member-identifier

The attributes are specified in Table 237.

Table 237 – Referencing MENUS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.26 Referencing METHODS of specific BLOCK_A instance**Purpose**

This referencing is used to access a member of the METHODS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, METHOD, member-identifier

The attributes are specified in Table 238.

Table 238 – Referencing METHODS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.27 Referencing COMPONENT instances**Purpose**

This referencing is used to reference an instance that is related with a COMPONENT_RELATION from a component. If the reference is used within a MENU declaration and the instance does not exist, the referenced item will be treated as invalid. If the reference is used in a METHOD DECLARATION and the instance does not exist, the METHOD will be aborted. Such a reference is not allowed to use within a conditional expression.

Lexical structure

reference, expression, identifier

The attributes of component referencing are specified in Table 239.

Table 239 – Referencing a COMPONENT instance

Usage	Attribute	Description
o	reference	Reference to a COMPONENT.
o	expression	Expression, which shall be evaluated to a non-negative integer that is within the index range of the COMPONENT_RELATION.
o	identifier	Identifier of a basic construct of the referenced COMPONENT.

7.38.28 Referencing COMPONENT types

Purpose

This referencing is used to reference a component type. All conditional expressions are evaluated with INITIAL_VALUES or if not defined, DEFAULT_VALUES or with data type defaults.

Lexical structure

reference, identifier

The attributes of component referencing are specified in Table 240.

Table 240 – Referencing a COMPONENT type

Usage	Attribute	Description
o	reference	Reference to a COMPONENT
o	identifier	Identifier of a basic construct of the referenced COMPONENT

7.38.29 Referencing FILES of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the FILES attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, FILE, member-identifier

The attributes are specified in Table 241.

Table 241 – Referencing FILES of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.38.30 Referencing PLUGINS of specific BLOCK_A instance

Purpose

This referencing is used to access a member of the PLUGINS attribute of a specific BLOCK_A instance.

Lexical structure

reference, expression, PLUGIN, member-identifier

The attributes are specified in Table 242.

Table 242 – Referencing PLUGINS of specific BLOCK_A instance

Usage	Attribute	Description
m	reference	Reference to a BLOCK_A instance.
m	expression	Non-negative integer that specifies the occurrence of the BLOCK_A instance within the device. If the specified block does not exist, the EDD application shall behave as if the referenced item's VALIDITY attribute is FALSE.
m	member-identifier	Identifier of an item of MEMBERS.

7.39 Strings

7.39.1 Specifying a string as a string literal

Purpose

A text string can be specified as a string literal (see A.2.3).

Lexical structure

string

The attribute is specified in Table 243.

Table 243 – String as a string literal

Usage	Attribute	Description
m	string	String literal.

7.39.2 Specifying a string as a string variable

Purpose

A text string specified as a string variable is the value of the string variable.

Lexical structure

reference

The attribute is specified in Table 244.

Table 244 – String as a string variable

Usage	Attribute	Description
m	reference	Reference to a VARIABLE instance.

7.39.3 Specifying a string as an enumeration value

Purpose

An enumeration value string is a string associated with one of the values of an enumeration variable.

Lexical structure

reference, value

The attributes are specified in Table 245.

Table 245 – String as an enumeration value

Usage	Attribute	Description
m	reference	Reference to a VARIABLE instance of TYPE ENUMERATED or BIT_ENUMERATED.
m	value	Corresponding value of the description text within the enumeration list.

EXAMPLE 1 The following enumeration value string specifies the string associated with the value 4 of the ENUMERATED variable units_code: units_code (4).

EXAMPLE 2 The following enumeration value string specifies the string associated with the value 0x02 of the BIT_ENUMERATED variable var_bitenum: var_bitenum (0x02).

7.39.4 Specifying a string as a dictionary reference

Purpose

A dictionary reference points to a string in a text dictionary.

Lexical structure

reference

The attribute is specified in Table 246.

Table 246 – String as a dictionary reference

Usage	Attribute	Description
m	reference	Reference to one or more strings specified in a dictionary (see 7.41).

7.39.5 Referencing HELP and LABEL attributes of EDD instances

Purpose

To get the current LABEL or the HELP string of an EDD instance, the instance and the LABEL or HELP attribute shall be referenced.

Lexical structure

reference, [HELP, LABEL]

The attributes are specified in Table 247.

Table 247 – Referencing HELP and LABEL attributes of EDD instances

Usage	Attribute	Description
m	reference	EDD instance which contains the attributes LABEL and/or HELP.
s	HELP	HELP string of the instance.
s	LABEL	LABEL string of the instance.

7.39.6 String operations

Purpose

String literals, string variables, string enumeration values, dictionary references and attribute values of basic constructs can be concatenated with one another. If a term contains references of numeric data types, the values shall be converted to a string by the EDD application.

EXAMPLE variable_of_type_string = "Description"
 "Electronic " + "Device " + variable_of_type_string results in "Electronic Device Description".

Lexical structure

(string)<exp>, [(concatenation-operator, string)<exp>]+

The attributes are specified in Table 248.

Table 248 – String operation

Usage	Attribute	Description
m	string	Either string literals, string or numeric data type variables, string enumeration values, dictionary references or references to string attributes, for example HELP or LABEL. The EDD application shall convert evaluated expressions and numeric data type variable to strings (see Clause A.3).
o	concatenation-operator	The strings shall be concatenated.

7.39.7 Prompt string formats

Purpose

Prompt strings can be use to embed formatted items into strings. These items are composed of an optional format specifier and a VARIABLE ID or reference.

Lexical structure

format, (reference, id)
 label, (reference, id)

The attributes are specified in Table 249.

Table 249 – Format specifier

Usage	Attribute	Description
m	format	Consists of the printf format string of the programming language-C (see ISO/IEC 9899). If the format is not specified, the format of the referenced VARIABLE instance is used. If no format is specified, a default format is used based on the type of VARIABLE instance.
m	label	LABEL of the VARIABLE instance.
o/s	reference	Value or label of the referenced VARIABLE instance.
o/s	id	Item IDs of the VARIABLES.

7.40 Expression

7.40.1 General structure

Purpose

There are three types of expressions.

- Primary expressions include constants, parenthesis-enclosed expressions, VARIABLE references, index element accesses or access to the status of VARIABLE instances. These operations are left-associative, i.e., they are evaluated from left to right.
- Unary expressions include all operators evaluating a single operand. These operations are right-associative, i.e., they are evaluated from right to left.
- Binary expressions include all operators evaluating two operands. These operations are left-associative, i.e., they are evaluated from left to right.

7.40.2 Primary expressions

Table 250 below summarizes the primary expressions in the EDDL.

Table 250 – Primary expressions

Primary expression	Description
constant	Expression whose value is identical to the value of the constant.
parenthesized expression	Expression whose value is identical to the value of the enclosed expression.
variable reference	Expression whose value is the value of the referenced VARIABLE.
ARRAY_INDEX	Expression whose value was taken from the index number of the referencing VALUE_ARRAY or LIST.
attribute values of VARIABLES	Reference consisting of the identifier of a VARIABLE instance and an attribute. The reference supplies the according value of the attribute (see Table 251 for the returned values).
PI	Mathematical constant π as a double.
attribute values of an AXIS	Reference to an AXIS instance and an attribute. The reference supplies the corresponding value of the attributes (see Table 252 for the returned values). These attributes may only be read unless specified otherwise.
attribute values of a BLOB	Reference to a BLOB instance and an attribute. The reference supplies the corresponding value of the attributes (see Table 253 for the returned values). These attributes may only be read unless specified otherwise.
attribute values of a LIST	Reference to a LIST instance and an attribute. The reference supplies the corresponding value of the attributes (see Table 254 for the returned values). These attributes may only be read unless specified otherwise.
attribute values of an ARRAY	Reference to an ARRAY instance and an attribute. The reference supplies the corresponding value of the attributes (see Table 255 for the returned values). These attributes may only be read unless specified otherwise.
LINEAR	Linear scaling.
LOGARITHMIC	Logarithmic scaling.
TRUE	Boolean value.
FALSE	Boolean value.
NULL	Unknown object or item.
FIRST	For LIST first element of the list. For OBJECT_REFERENCE, first sibling object in the object hierarchy.
LAST	For LIST last element of the list. For OBJECT_REFERENCE, navigation to the last sibling object in the object hierarchy.
CHILD	For OBJECT_REFERENCE, navigation to the first descendent object.
PARENT	For OBJECT_REFERENCE, navigation to the next higher hierarchy level.
NEXT	For OBJECT_REFERENCE, navigation to the next sibling object.
PREV	For OBJECT_REFERENCE, navigation to the previous sibling object.
SELF	For OBJECT_REFERENCE, navigation to own object ("this").

Table 251 – Attribute values of VARIABLES

Attributes	Data type	Description
MIN_VALUE	type of VARIABLE instance	Value which complies with the MIN_VALUE of the VARIABLE instance. If more than one MIN_VALUE is defined, MIN_VALUE plus the range identifier (see Table 168) shall be used for referencing. If no MIN_VALUE is defined, the data type specific minimum value (see Table D.9) should be used in the evaluation.
MAX_VALUE	type of VARIABLE instance	Value which complies with the MAX_VALUE of the VARIABLE instance. If more than one MAX_VALUE is defined, MAX_VALUE plus the range identifier (see Table 168) shall be used for referencing. If no MAX_VALUE is defined, the data type specific maximum value (see Table D.9) should be used in the evaluation.
SCALING_FACTOR	double	Value which complies with the SCALING_FACTOR of the VARIABLE instance.
DEFAULT_VALUE	type of VARIABLE instance	Value which complies with the DEFAULT_VALUE of the VARIABLE instance. If no DEFAULT_VALUE is defined, the data type specific initial value (see Table D.9) should be used in the evaluation. In expressions with DEFAULT_VALUE the variable attribute is always referenced, even if COMPONENT INITIAL_VALUES or TEMPLATE DEFAULT_VALUES are defined.
INITIAL_VALUE	type of VARIABLE instance	Value which complies with the INITIAL_VALUE of the VARIABLE instance. If no INITIAL_VALUE is defined, the data type specific initial value (see Table D.9) should be used in the evaluation.
VARIABLE_STATUS		Value which indicates the status of VARIABLE. The following statuses are defined: ▪ VARIABLE_STATUS_NONE specifies that the value is valid and not one of the other status; ▪ VARIABLE_STATUS_INVALID specifies that the value is out of range; ▪ VARIABLE_STATUS_NOT_ACCEPTED specifies that the value could not be transferred; ▪ VARIABLE_STATUS_NOT_SUPPORTED specifies that the value is not supported; ▪ VARIABLE_STATUS_CHANGED specifies that the value has been changed by the user or by a method; ▪ VARIABLE_STATUS_LOADED specifies that the value has been successfully loaded; ▪ VARIABLE_STATUS_INITIAL specifies that the value is unchanged; ▪ VARIABLE_STATUS_NOT_CONVERTED specifies that the value has not been converted, for example, because the units are not compatible.

Table 252 – AXIS attribute values

Attributes	Data type	Description
MIN_VALUE	arithmetic type	Value which complies with the MIN_VALUE of the AXIS instance.
MAX_VALUE	arithmetic type	Value which complies with the MAX_VALUE of the AXIS instance.
SCALING	LINEAR or LOGARITHMIC	Value which complies with the SCALING of the AXIS instance.
VIEW_MIN	arithmetic type	Value which complies with the VIEW_MIN of the AXIS instance; VIEW_MIN can be read and written.
VIEW_MAX	arithmetic type	Value which complies with the VIEW_MAX of the AXIS instance; VIEW_MAX can be read and written.

Table 253 – BLOB attribute values

Attributes	Data type	Description
COUNT	unsigned_integer	Value which complies with the COUNT of the BLOB instance. The size of the BLOB is returned, i.e. the number of bytes.
IDENTITY	string	Value which complies with the IDENTITY of the BLOB instance. The value will be an empty string if the IDENTITY attribute is not specified and the browse identity Builtin has not been called.

Table 254 – LIST attribute values

Attributes	Data type	Description
CAPACITY	unsigned_integer	Value which complies with the CAPACITY of the LIST instance.
COUNT	unsigned_integer	Value which complies with the COUNT of the LIST instance.

Table 255 – ARRAY attribute values

Attributes	Data type	Description
NUMBER_OF_ELEMENTS	integer	Value which complies with the NUMBER_OF_ELEMENTS of the VALUE_ARRAY instance.

7.40.3 Unary expressions

A unary expression consists of a numerical operand, an expression, preceded by a unary operator. Table 256 specifies the unary expressions in the EDDL.

Table 256 – Unary expressions

Operator	Description
-	Arithmetic negation of its operand.
~	Bitwise negation of its operand, that is, each bit of the result is the inverse of the corresponding bit of the operand. The operand of the ~ operator shall have an integral value.
!	Logical negation of its operand.

7.40.4 Binary expressions

7.40.4.1 General structure

A binary expression consists of two operands or expressions, separated by a binary operator. If either operand has a floating-point value, the other operand is converted (promoted) to a floating-point value. Subclause 7.40.4 specifies the following types of binary expressions.

- Multiplicative
- Additive
- Shift
- Relational
- Equality
- Bitwise AND (&)
- Bitwise XOR (^)
- Bitwise OR (|)
- Logical AND (&&)
- Logical OR (||)
- Conditional evaluation

7.40.4.2 Multiplicative operators

Multiplicative operators specify the multiplication and division of operands. Table 257 specifies the multiplicative operators.

Table 257 – Multiplicative operators

Operator	Description
*	Multiplication of its operands.
/	Division of the first operand by the second operand. The result of the / operator is the quotient of the division.
%	Division of the first operand by the second operand. The result of the % operator is the remainder.

7.40.4.3 Additive operators

Additive operators specify the addition and subtraction of numerical operands. If this operator is applied on numerical operands, then the result is the sum of its two operands, and the operation is commutative. If the + operator is applied on string operands, then the result is a string value generated by appending the second string to the first string, and the operation is not commutative.

The minus operator indicates a subtraction. The resulting value of this operation is the difference of its operands. The second operand is subtracted from the first. Table 258 specifies the additive operators.

Table 258 – Additive operators

Operator	Description
+	Addition of its operands.
-	Subtraction of the second operand from the first.

7.40.4.4 Shift operators

The << and >> operators specify a shift of the first operand by the number of bits specified by the second operand. The operands of the << and >> operators shall have integer values. Table 259 specifies the shift operators.

Table 259 – Shift operators

Operator	Description
<<	The first operand to the left. The bits shifted off are discarded, and the vacated bits are zero filled.
>>	The first operand to the right. The bits shifted off are discarded. If the first operand is less than 0, the vacated bits are one filled; otherwise they are zero filled.

7.40.4.5 Relational operators

Relational operators (<, <=, >, >=) specify a comparison of its operands. The result of this type of expression is 1 if the tested relationship is true; otherwise, the result is 0. The operands shall be numerical data types. Table 260 specifies the relational operators.

Table 260 – Relational operators

Operator	Description
<	Tests for the relationship "less than".
<=	Tests for the relationship "less than or equal to".
>	Tests for the relationship "greater than".
>=	Tests for the relationship "greater than or equal to".

7.40.4.6 Equality operators

The equality operators are == and !=. The result of this type of expression is 1 if the tested relationship is true; otherwise, the result is 0. The operators shall be of the same data type. Any data types are allowed. Table 261 specifies the equality operators.

Table 261 – Equality operators

Operator	Description
==	Tests for the relationship "equals".
!=	Tests for the relationship "does not equal".

7.40.4.7 Bitwise AND operator (&)

The & operator specifies the bitwise AND of its integer operands, that is, each bit of the result is set if each of the corresponding bits of the operands is set. The operands of the & operator shall have integral values.

7.40.4.8 Bitwise XOR operator (^)

The ^ operator specifies the bitwise exclusive OR of its integer operands, that is, each bit of the result is set if only one of the corresponding bits of the operands is set. The operands of the ^ operator shall have integral values.

7.40.4.9 Bitwise OR operator (|)

The | operator specifies the bitwise inclusive OR of its integer operands, that is, each bit of the result is set if either of the corresponding bits of the operands is set. The operands of the | operator shall have integral values.

7.40.4.10 Logical AND operator (&&)

The && operator specifies the Boolean AND evaluation of its integer operands. If either operand references an invalid VARIABLE (VALIDITY equal to FALSE), the operand shall evaluate to 0. The result of this type of expression is 1 if both of the operands are not equal to 0; otherwise, the result is 0. If the first operand is equal to 0, the second operand is not evaluated.

7.40.4.11 Logical OR operator (||)

The || operator specifies the Boolean OR evaluation of its integer operands. If either operand references an invalid VARIABLE (VALIDITY equal to FALSE), the operand shall evaluate to 0. The result of this type of expression is 1 if either of the operands is not equal to 0; otherwise, the result is 0. If the first operand is not equal to 0, the second operand is not evaluated.

7.40.4.12 Conditional evaluation

The operators ? and: are used for conditional evaluation.

EXAMPLE 1

```
MIN_VALUE (var1 > 0) ? 10: 0;
```

The logical expression is evaluated and if it is non-zero, the MIN_VALUE attribute is 10; otherwise 0.

EXAMPLE 2 The following if-else structure is logically identical to example 1:

```
MIN_VALUE IF (var1 > 0) { 10; } ELSE { 0; }
```

7.41 Text dictionary

General structure

The dictionary is a collection of text strings that can be used in an EDD. The dictionary text is referenced within an EDD. The dictionary can be accessed from any EDD to assure consistency of text strings (for example, used for LABEL and HELP attributes).

Generally, one common text dictionary is used for a group of device types. Additionally, a user specific dictionary can be specified. The text dictionary supports multilingual texts.

Lexical structure

```
section-number, string-number identifier (language-code, country-code,  
string)+
```

The attributes are specified in Table 262.

Table 262 – Text dictionary attributes

Usage	Attribute	Description
m	section-number	Definition of a section inside a code table. If no code table is used, this value is '0'.
m	string-number	Definition of an offset of the dictionary entry inside a code table. If no code table is used, this value is '0'.
m	identifier	Name of a text string in an EDD. The identifier shall be unique in a combination of section-number and string-number for all used dictionaries. Identifiers shall be unique except the use of 'xxx_highoffset' to terminate a section.
o	language-code	Language used and the ASCII or multi code presentation of a string; language codes are defined in ISO 639 (see A.2.4).
o	country-code	Country in which the language is spoken; country codes are defined in ISO 3166-1. Country code can only be specified if language code is defined.
m	string	Text string in the related language.

String literals are defined in A.2.4.

7.42 PLUGIN

7.42.1 General structure

Purpose

PLUGIN describes a User Interface Plug-in (UIP).

Lexical structure

PLUGIN identifier [HELP, LABEL, UUID, VALIDITY, VISIBILITY]

The attributes of PLUGIN are specified in Table 263.

Table 263 – PLUGIN attributes

Usage	Attribute	Description
m	LABEL	Lexical element (see 7.36.9).
m	UUID	Lexical element (see 7.42.2).
o	HELP	Lexical element (see 7.36.8).
o	VALIDITY	Lexical element (see 7.36.16).
o	VISIBILITY	Lexical element (see 7.36.19).

7.42.2 Specific attribute – UUID

Purpose

The UUID attribute specifies the unique identifier used to identify the PLUGIN. The UUID attribute is a universally unique identifier as specified in ISO/IEC 9834-8.

NOTE This attribute may not be conditional.

Lexical structure

UUID uuid

The attribute of UUID is specified in Table 264.

Table 264 – UUID attribute

Usage	Attribute	Description
m	uuid	Unique identifier of the plug-in specified in the manifest.

7.43 BLOB

Purpose

The Binary Large Object (BLOB) shall be used to transfer binary data to or from a device. The EDD application has read-only access to contents of a BLOB, for example for validation purposes only. A BLOB is a data container whose content is accessible via Builtins.

Lexical structure

BLOB identifier [HANDLING, HELP, IDENTITY, LABEL]

The attributes of BLOB are specified in Table 265.

Table 265 – BLOB attributes

Usage	Attribute	Description
m	HANDLING	Lexical element (see 7.36.6). NOTE HANDLING READ means the BLOB can only be read from the device
o	HELP	Lexical element (see 7.36.8).
o	IDENTITY	Lexical element (see 7.36.21). If IDENTITY is not defined, the filename shall be selected using a method (see IEC 61804-5).
o	LABEL	Lexical element (see 7.36.9).

Annex A (normative)

EDDL formal definition

A.1 EDDL preprocessor

A.1.1 General structure

Purpose

The preprocessor generates an EDD before compilation. The preprocessor replaces parts of the EDD text, inserts the contents of other files or suppresses the processing of parts of the EDD by removing sections.

Structure

All preprocessor directives start with a hash symbol (#) as the first character on a line. A space (SPACE or TAB characters) can appear after the initial # for proper indentation.

A.1.2 Directives

A.1.2.1 #define

Purpose

The #define directive gives a meaningful name to a constant in the EDD.

Structure

```
#define identifier token-string
```

This form of the #define directive instructs the preprocessor to replace all subsequent occurrences of the identifier in the EDD file with token-string.

NOTE The identifier is not replaced if it appears in a comment, within a string, or as part of a longer identifier.

```
#define identifier
```

This form of the #define directive instructs the preprocessor to remove all occurrences of the identifier from the EDD file. The identifier remains defined and can be tested using the #if defined and #ifdef directives.

```
#define identifier (argument, ... ,argument) token-string
```

This form of the #define directive instructs the preprocessor to create function-like macros. This form accepts a list of arguments that shall appear in parentheses and are separated by commas.

When a macro has been defined in this syntax form, subsequent textual instances followed by an argument list constitute a macro call. The actual arguments following an instance of identifier in the source file are matched to the corresponding formal parameters in the macro definition. Each formal parameter in token-string is replaced by the corresponding actual argument. Any macros in the actual argument are expanded before the directive replaces the formal parameter.

Each argument in the list shall be unique. No spaces are allowed to separate identifier and the opening parenthesis.

For long directives on multiple source lines, a backslash (\) shall be used before the

NEWLINE character.

No spaces are allowed between name and the “(“.

A.1.2.2 **#include**

Purpose

The **#include** directive tells the preprocessor to treat the contents of a specified file as if those contents had appeared in the source program at the point where the directive appears.

Structure

```
#include "filename"
#include <filename>
```

Read the contents of the file named with filename. The preprocessor processes this data as if it was part of the current file.

Both syntax forms cause replacement of that directive by the entire contents of the specified include file. The difference between the two forms is the order in which the preprocessor searches for include files when the path is incompletely specified.

No additional tokens are permitted on the directive line after the final “ or >.

A.1.2.3 **#line**

Purpose

The **#line** directive causes the preprocessor to change the compiler's internally stored line number and filename to a given line number and filename. The EDD processor uses the line number and filename to refer to errors that it finds during processing. The line number usually refers to the current input line, and the filename refers to the current input file. The line number is incremented after each line is processed.

Structure

```
#line integer-constant "filename"
```

integer-constant and filename cause the preprocessor to substitute the internally stored line number of the EDD processor to a given line number and filename. integer-constant is interpreted as the line number of the next line and is incremented after each line is processed.

A.1.2.4 **#if, #elif, #else, and #endif**

Purpose

The **#if** directive, with the **#elif**, **#else**, and **#endif** directives, controls the processing parts of an EDD file.

```
#if constant-expression
```

Subsequent lines up to the **#else**, **#elif** or **#endif** directive, appear in the output only if constant-expression yields a non-zero value. The binary operators “&&” and “||” are legal in constant-expression as well as the “?:” operator and the unary “-”, “!” and “~” operators.

In addition the unary operator defined can be used in special constant expressions, as shown by the following syntax:

```
defined(identifier)
defined identifier
```

This constant expression is considered true (non-zero) if the identifier is currently defined; otherwise, the condition is false. An identifier defined as empty text is considered defined. The defined directive can be used in an `#if` and an `#elif` directive, but nowhere else.

`#elif constant-expression`

Any number of `#elif` directives may appear between an `#if`, `#ifdef` or `#ifndef` directive and a matching `#else` or `#endif` directive. The lines following the `#elif` directive appear in the output only if all the following conditions hold:

- The constant-expression in the preceding `#if` directives is evaluated to zero, the name in the preceding `#ifdef` is not defined or the name in the preceding `#ifndef` directives was defined.
- The constant-expressions in all inverting `#elif` directives are evaluated to zero.
- The current constant-expression evaluates to non-zero.

If the constant-expression evaluates to non-zero, ignore subsequent `#elif` and `#else` directives up to the matching `#endif`. Any constant-expression allowed in an `#if` directive is allowed in an `#elif` directive.

`#else`

If all occurrences of constant-expression are false, or if no `#elif` directives appear, the preprocessor selects the text block after the `#else` clause. If the `#else` clause is omitted and all instances of constant-expression in the `#if` block are false, no text block is selected.

`#endif`

Ends a section of lines which starts with one of the conditionals directives `#if`, `#ifdef` or `#ifndef`. Each directive shall have a matching `#endif`.

NOTE 1 The preprocessor recognizes formal parameters for macros in `#define` directive bodies, even when they occur outside character constants and quoted strings. Macro names are not recognized within character constants or quoted strings during the regular scan. Thus `#define aaa bbb` does not expand `aaa` into LABEL "aaa".

NOTE 2 Macros are not expanded while processing a `#define` or `#undef`. Thus:

```
#define aaa bbb
#define ccc aaa
#undef aaa
ccc
produces aaa.
```

NOTE 3 Macros are not expanded during the scan which determines the actual parameters to another macro call. Thus:

```
#define turn(aaa,bbb) bbb aaa
#define ccc ddd
turn(ccc, #define ccc eee)
produces #define ddd eee ddd.
```

A.1.2.5 **#ifdef, #ifndef and #undef**

Purpose

The **#ifdef** and **#ifndef** directives control the processing parts of an EDD file. The **#ifdef** and **#ifndef** directives perform the same task as the **#if** directive when it is used with a defined identifier.

#ifdef identifier

Subsequent lines up to the matching **#else**, **#elif** or **#endif** appear in the output only if the identifier has been defined. The identifier is specified using either a **#define** directive or outside the EDD and in the absence of an intervening **#undef** directive. No additional tokens are permitted on the directive line after name.

#ifndef identifier

Subsequent lines up to the matching **#else**, **#elif** or **#endif** appear in the output only if the name has not been defined or if its definition has been removed with an **#undef** directive. No additional tokens are permitted on the directive line after the name.

#undef identifier

The **#undef** directive causes an identifier's preprocessor definition to be removed. No additional tokens are permitted on the directive line after the name.

A.1.3 **Predefined macros**

A.1.3.1 **General structure**

The preprocessor shall recognize a specified set of predefined macros. These macros should be used anywhere in the EDD and are for informational purpose.

These macros take no arguments and cannot be redefined.

A.1.3.2 **List of predefined macros**

A.1.3.2.1 **__FILE__**

Purpose

__FILE__ contains the name of the current EDD file.

Structure

__FILE__ expands to a string surrounded by double quotation marks.

A.1.3.2.2 **__LINE__**

Purpose

__LINE__ contains the line number in the current EDD file.

Structure

__LINE__ is a decimal integer constant. It can be altered with a **#line** directive.

A.1.4 NEWLINE characters

A NEWLINE character terminates a character constant or a quoted string. The backslash (\) followed by a NEWLINE in the body of a #define statement is used to continue the definition onto the next line.

A.1.5 Comments

A comment is a sequence of characters beginning with a forward slash/asterisk combination (/*) and ends with an asterisk/slash combination (*). A comment can include any combination of characters from the representable character set, including NEWLINE characters.

Furthermore, the single-line comment is supported if it is preceded by two forward slashes (//). A comment beginning with two forward slashes is terminated by the next NEWLINE character that is not preceded by an escape character.

A.2 Conventions

A.2.1 Integer constants

An integer constant may be specified in binary, octal, decimal, or hexadecimal notation. Table A.1 specifies the conventions for each type of notation.

Table A.1 – Conventions for integer constants

Integer Constant Type	Conventions
binary	Non-empty sequence of the binary digits 0 and 1 preceded by either 0b or 0B.
octal	Non-empty sequence of the digits 0 through 7 beginning with 0.
decimal	Non-empty sequence of the decimal digits 0 through 9, not beginning with 0.
hexadecimal	Non-empty sequence of hexadecimal digits preceded by either 0x or 0X. The hexadecimal digits are the digits 0 through 9 and the letters a through f (or A through F) with the values 10 through 15, respectively.

A.2.2 Floating-point constants

Lexical structure

A floating-point constant has four parts:

- an integer part, a sequence of decimal digits;
- a decimal point (.);
- a fraction part, a sequence of decimal digits;
- an exponent part, a possibly signed sequence of decimal digits preceded by one of the letters e or E.

Rules

The following rules apply to using floating-point constants:

- either the integer part or the fraction part can be omitted, but not both;
- either the decimal point or the exponent part can be omitted, but not both.

Example: Following are examples of floating-point constants:

59,
,87

48,93

4,8e12

The calculation of the algebraic expression shall use the most precise data type.

A.2.3 String literals

String literals in an EDD shall be encoded using either ISO Latin-1 (ISO/IEC 8859-1) or UTF-8 (RFC 3629). Within a given EDD all string literals shall use the same encoding. A string literal is a possibly empty sequence of characters enclosed in double quotes ("). The enclosed characters shall be any character except the following:

- double quote (");
- backslash (\);
- new line.

Using escape sequences in string literals a string constant can also contain escape sequences that represent a character. Table A.2 shows the escape sequences and their result.

Table A.2 – Using escape sequences in string literals

Escape code	Result
'	Single quote.
"	Double quote.
	Vertical bar.
?	Question mark.
\	Backslash.
\a	Alert.
\f	Form feed.
\n	New line.
\r	Carriage return.
\t	Horizontal tab.
\v	Vertical tab.

A.2.4 Using language and country codes in string literals

A string literal can encapsulate all the translations of a given phrase. The language code is specified using ISO/IEC 8859-1. If a string literal does not contain translations for all the languages, a default language shall be used. Table A.3 shows examples of language codes.

Table A.3 – Language code examples for string literals

Language code	Language
no code assigned	Default language, used if a language is requested that is not defined.
zh	Chinese
de	German
en	English
es	Spanish
fr	French
it	Italian
ja	Japanese
ko	Korean

English shall be the default language.

EXAMPLE 1 The following string literal specifies the English phrase "Invalid Selection" in English and German:

"Invalid Selection"

"|de|Unzulässige Auswahl"

In addition to the standard country codes defined in ISO 3166-1, the zz country code may be used to specify a short version of a string. Short versions of strings are useful for EDD applications that run on mobile devices that have a limited screen size.

EXAMPLE 2

write_protected

"Write Protected"

"|de|Schreibgeschützt"

"|fr|Protégé en écriture"

"|it|Protetto in scrittura"

"|es|Protegido contra escritura"

"|en zz|Wr Prot"

"|de zz|Geschützt"

The identifier "write_protected" is the identifier which is referenced within the EDD. The qualifiers |de|, |fr|, |it|, |es| specify the text for the languages German, French, Italian and Spanish. The "en zz" and "de zz" qualifiers specify short versions of the English and German strings. This same approach could also be used to provide short versions for French, Italian, and Spanish.

A.3 Operators

Table A.4 defines the precedence and associativity of EDDL operators.

Table A.4 – Precedence and associativity for EDDL operators

Precedence	Associativity	Operator	Supported outside METHODS	Description	Category
1 (highest)	Left-to-right	++	No	Suffix increment	Arithmetic
1	Left-to-right	--	No	Suffix decrement	Arithmetic
1	Left-to-right	()	Yes	Function call	Function call
1	Left-to-right	[]	Yes	Array indexing	Referencing
1	Left-to-right	.	Yes	Referencing of sub items and	Referencing
1	Left-to-right	->	Yes	Object referencing	Referencing
2	Right-to-left	++	No	Prefix increment	Arithmetic
2	Right-to-left	--	No	Prefix decrement	Arithmetic
2	Right-to-left	+	Yes	Unary plus	Arithmetic
2	Right-to-left	-	Yes	Unary minus	Arithmetic
2	Right-to-left	!	Yes	Logical NOT	Arithmetic
2	Right-to-left	~	Yes	Binary NOT	Binary
3	Left-to-right	*	Yes	Arithmetic multiplication	Arithmetic
3	Left-to-right	/	Yes	Arithmetic division	Arithmetic
3	Left-to-right	%	Yes	Modulo (remainder)	Arithmetic
4	Left-to-right	+	Yes	Arithmetic addition or string concatenation	Arithmetic, string
4	Left-to-right	-	Yes	Arithmetic subtraction	Arithmetic
5	Left-to-right	<<	Yes	Left shift	Binary
5	Left-to-right	>>	Yes	Right shift	Binary
6	Left-to-right	<	Yes	Less than	Comparison
6	Left-to-right	<=	Yes	Less than or equal	Comparison
6	Left-to-right	>	Yes	Greater than	Comparison
6	Left-to-right	>=	Yes	Greater than or equal to	Comparison
7	Left-to-right	==	Yes	Equal	Comparison
7	Left-to-right	!=	Yes	Not equal	Comparison
8	Left-to-right	&	Yes	Binary and	Binary
9	Left-to-right	^	Yes	Binary XOR (exclusive or)	Binary
10	Left-to-right		Yes	Binary OR	Binary
11	Left-to-right	&&	Yes	Logical AND	Arithmetic
12	Left-to-right		Yes	Logical OR	Arithmetic
13	Right-to-left	?:	Yes	Ternary condition	Conditional
13	Right-to-left	=	No	Direct assignment	Assignment

Precedence	Associativity	Operator	Supported outside METHODs	Description	Category
13	Right-to-left	+=	No	Assignment by sum	Assignment
13	Right-to-left	-=	No	Assignment by difference	Assignment
13	Right-to-left	*=	No	Assignment by product	Assignment
13	Right-to-left	/=	No	Assignment by quotient	Assignment
13	Right-to-left	%=	No	Assignment by modulo (remainder)	Assignment
13	Right-to-left	<<=	No	Assignment by bitwise left shift	Assignment
13	Right-to-left	>>=	No	Assignment by bitwise right shift	Assignment
13	Right-to-left	&=	No	Assignment by bitwise and	Assignment
13	Right-to-left	^=	No	Assignment by bitwise XOR	Assignment
13	Right-to-left	=	No	Assignment by bitwise or	Assignment
14	Left-to-right	,	No	Separation of expressions	Statement

Table A.5 defines the permitted operations for VARIABLES or METHOD local variables according to their data types.

Table A.5 – Operations for VARIABLES or METHOD local variables

VARIABLE TYPE	METHOD local data type	Operators
BOOLEAN	N.a.	== != && ! =
DOUBLE, FLOAT	double, float	+ – * / < <= == != >= > = += -= *= /=
INTEGER, UNSIGNED_INTEGER	char, int, long, long long, unsigned char, unsigned int, unsigned long, unsigned long long	+ – * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
DATE	N.a.	== != < <= >= > =
DATE_AND_TIME	N.a.	== != < <= >= > =
DURATION	N.a.	== != < <= >= > =
TIME	N.a.	== != < <= >= > =
TIME_VALUE(8)	N.a.	== != < <= >= > =
TIME_VALUE(4)	N.a.	== != < <= >= > =
BIT_ENUMERATED	unsigned char, unsigned int, unsigned long, unsigned long long	+ – * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
ENUMERATED	unsigned char, unsigned int, unsigned long, unsigned long long	+ – * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
INDEX	unsigned char, unsigned int, unsigned long, unsigned long long	+ – * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
OBJECT_REFERENCE	N.a.	== != = ->
ASCII, PACKED_ASCII, PASSWORD, VISIBLE	DD_STRING char []	+ == != [] =
BITSTRING	unsigned char[]	== !=
EUC	DD_STRING char []	+ == != [] =

VARIABLE TYPE	METHOD local data type	Operators
OCTET	unsigned char[]	+ == != [] =
N.a.	time_t	== != =
Key: N.a.: not applicable		

Concatenation of string literals without '+' operator shall be also supported (see A.6.36, Common attributes "string_literal:").

A.4 Keywords

Table A.6 contains the EDDL keywords.

Table A.6 – EDDL keywords

EDDL keyword	EDDL keyword	EDDL keyword
ACCESS	ACTUATOR	ACTUATOR_ELECTRIC
ACTUATOR_ELECTRO_PNEUMATIC	ACTUATOR_HYDRAULIC	ADD
ADDRESSING	ALARM	ALIGN_LEFT
ALIGN_RIGHT	ANALOG_INPUT	ANALOG_OUTPUT
AO	AO1	AO2
AO3	AO4	AO5
API	ARRAY	ARRAY_INDEX
ARRAYS	ASCII	AUTO_CREATE
AXES	AXIS	AXIS_ITEMS
BAD	BIT_ENUMERATED	BITSTRING
BLOB	BLOBS	BLOCK
BLOCKS	BOOLEAN	break
BYTE_ORDER		
CAN_DELETE	CAPACITY	case
CASE	CATALOG	char
CHARACTERISTICS	CHART	CHART_ITEMS
CHARTS	CHECK_CONFIGURATION	CHILD
CHILD_COMPONENT	CLASS	CLASSIFICATION
COLLECTION	COLLECTION_ITEMS	COLLECTIONS
COLUMNBREAK	COMM_ERROR	COMMAND
COMMANDS	COMMANDS	COMPONENT
COMPONENT_FOLDER	COMPONENT_FOLDERS	COMPONENT_PARENT
COMPONENT_PATH	COMPONENT_REDEFINITIONS	COMPONENT_REFERENCE
COMPONENT_REFERENCES	COMPONENT_RELATION	COMPONENT_RELATIONS
COMPONENTS	COMPUTATION	CONNECTION_POINT
CONSTANT_UNIT	CONTAINED	continue
CONTROLLER	CONVERTER	CORRECTABLE

EDDL keyword	EDDL keyword	EDDL keyword
CORRECTION	COUNT	CYCLE_TIME
DATA	DATA_ENTRY_ERROR	DATA_ENTRY_WARNING
DATA1	DATA2	DATA3
DATA4	DATA5	DATA6
DATA7	DATA8	DATA9
DATE	DATE_AND_TIME	DD_ITEM
DD_REVISION	DD_STRING	DECLARATION
default	DEFAULT	DEFAULT_VALUE
DEFAULT_VALUES	DEFINITION	DELETE
DETAIL	DETECT	DEVICE
DEVICE_REVISION		DEVICE_TYPE
DIAGNOSE	DIAGNOSTIC	DIALOG
DIGITAL_INPUT	DIGITAL_OUTPUT	DISCRETE
DISCRETE_IN	DISCRETE_INPUT	DISCRETE_OUT
DISCRETE_OUTPUT	DISPLAY_FORMAT	DISPLAY_ITEMS
double	DOUBLE	DURATION
DV	DV1	DV2
DV3	DV4	DYNAMIC
EDD	EDD_PROFILE	EDD_REVISION
EDD_VERSION	EDIT_DISPLAY	EDIT_DISPLAY_ITEMS
EDIT_DISplays	EDIT_FORMAT	EDIT_ITEMS
ELECTRICAL_DISTRIBUTION	ELECTRICAL_DISTRIBUTION_POWER_MONITORING	ELEMENTS
else	ELSE	EMPHASIS
ENUMERATED	ERROR	EUC
EVENT	EVERYTHING	EXIT_ACTIONS
FALSE	FF	FILE
FILE_ITEMS	FILES	FILTER
FIRST	float	FLOAT
for	FREQUENCY	FREQUENCY_CONVERTER
FREQUENCY_INPUT	FREQUENCY_OUTPUT	FUNCTION
GAUGE	GRAPH	GRAPH_ITEMS
GRAPHS	GRID	GRID_ITEMS
GRIDS	GROUP	
HANDLING	HARDWARE	HART
HEADER	HEIGHT	HELP
HIDDEN	HIGH_HIGH_LIMIT	HIGH_LIMIT
HORIZONTAL	HORIZONTAL_BAR	HOURS
IDENTITY	if	IF
IGNORE_IN_HANDHELD	IGNORE_IN_HOST	IGNORE_IN_TEMPORARY_MASTER
IMAGE	IMAGE_ITEMS	IMAGES
IMPORT	INDEX	INDICATOR
INFO	INIT_ACTIONS	INITIAL_VALUE
INITIAL_VALUES	INLINE	INPUT

EDDL keyword	EDDL keyword	EDDL keyword
int	INTEGER	INTERFACE
INTERFACES	IS_CONFIG	ITEM_ARRAY
ITEM_ARRAY_ITEMS	ITEM_ARRAYS	ITEMS
KEY_POINTS		
LABEL	LARGE	LAST
LENGTH	LIKE	LINE_COLOR
LINE_TYPE	LINEAR	LINK
LIST	LIST_ITEMS	LISTS
LOAD_TO_APPLICATION	LOAD_TO_DEVICE	LOCAL
LOCAL_DISPLAY	LOCAL_PARAM	LOCAL_PARAMETERS
LOGARITHMIC	long	LOW_LIMIT
LOW_LOW_LIMIT		
MANUFACTURER	MANUFACTURER_EXT	MARGINAL
MAX_VALUE	MAX_VALUE1	MAX_VALUE2
MAX_VALUE3	MAX_VALUE4	MAX_VALUE5
MAX_VALUE6	MAX_VALUE7	MAX_VALUE8
MAX_VALUE9	MIN_VALUE	MIN_VALUE1
MIN_VALUE2	MIN_VALUE3	MIN_VALUE4
MIN_VALUE5	MIN_VALUE6	MIN_VALUE7
MIN_VALUE8	MIN_VALUE9	MINIMUM_NUMBER
MINUTES	MISC	MISC_ERROR
MISC_WARNING	MODE	MODE_AUTOMATIC
MODE_CASCADE	MODE_ERROR	MODE_ERROR
MODE_INITIALIZATION	MODE_LOCAL_OVERRIDE	MODE_MANUAL
MODE_OUT_OF_SERVICE	MODE_REMOTE_CASCADE	MODE_REMOTE_OUTPUT
MORE		
NETWORK	NETWORK_COMMUNICATION_SERVICE_PROVIDER	NETWORK_COMPONENT
NETWORK_COMPONENT	NETWORK_COMPONENT_WIRELESSADAPTER	NETWORK_CONNECTION_POINT
NEXT	NEXT_COMPONENT	NEXT_COMPONENT
NO_LABEL	NO_UNIT	NUMBER
NUMBER_OF_ELEMENTS	NUMBER_OF_POINTS	
OBJECT_REFERENCE	OCTET	OF
OFFLINE	ON_UPDATE_ACTIONS	ONLINE
OPERATE	OPERATION	OPTIONAL
ORIENTATION	OUTPUT	
PACKED_ASCII	PAGE	PARAM
PARAM_LIST	PARAMETER_LISTS	PARAMETERS
PARENT	PARENT_COMPONENT	PASSWORD
PATH	PHYSICAL	PI
PLUGIN	PLUGIN_ITEMS	PLUGINS
POST_EDIT_ACTIONS	POST_READ_ACTIONS	POST_READSTRECEIVE_ACTIONS
POST_READSTUPDATE_ACTIONS	POST_USERCHANGE_ACTIONS	POST_WRITE_ACTIONS
PRE_EDIT_ACTIONS	PRE_READ_ACTIONS	PRE_WRITE_ACTIONS

EDDL keyword	EDDL keyword	EDDL keyword
PREV	PREV_COMPONENT	PRIVATE
PROCESS	PROCESS_ERROR	PROCESS_VALUE
PROFIBUS_DP	PROFIBUS_PA	PROFINET
PROTOCOL		
READ	READ_ONLY	RECORD
RECORDER	RECORDS	REDEFINE
REDEFINITIONS	REDUNDANCY	REFRESH
REFRESH_ACTIONS	REFRESH_ITEMS	REFRESHES
RELATION_TYPE	RELATIONS	REMOTEIO
REPLY	REQUEST	REQUIRED_INTERFACE
REQUIRED_RANGES	RESPONSE_CODES	return
REVIEW	ROWBREAK	
SCALING	SCALING_FACTOR	SCAN
SCAN_LIST	SCOPE	SECONDS
SELECT	SELF	SELF_CORRECTING
SEMICONDUCTOR	SENSOR	SENSOR_ACOUSTIC
SENSOR_ANALYTIC	SENSOR_ANALYTIC_GAS	SENSOR_ANALYTIC_LIQUID
SENSOR_CONCENTRATION	SENSOR_DENSITY	SENSOR_FLOW
SENSOR_FLOW_CORIOLIS	SENSOR_FLOW_ELECTRO_MAGNETIC	SENSOR_FLOW_MECHANICAL
SENSOR_FLOW_RADIOMETRIC	SENSOR_FLOW_THERMAL	SENSOR_FLOW_ULTRASONIC
SENSOR_FLOW_VORTEX_COUNTER	SENSOR_LEVEL	SENSOR_LEVEL_BUOYANCY
SENSOR_LEVEL_CAPACITIVE	SENSOR_LEVEL_ECHO	SENSOR_LEVEL_HYDROSTATIC
SENSOR_LEVEL_RADIOMETRIC	SENSOR_MULTIVARIABLE	SENSOR_POSITION
SENSOR_PRESSURE	SENSOR_TEMPERATURE	SEPARATOR
SERVICE	SHARED	short
SIBLING_COMPONENT	signed	SIMULATION
SLOT	SMALL	SOFTWARE
SOURCE	SOURCE_ITEMS	SOURCES
SPECIALIST	STATE	STRIP
STYLE	SUB_SLOT	SUCCESS
SUMMARY	SUPPLIED_INTERFACE	SWEEP
switch	SWITCHGEAR	
TABLE	TEMPLATE	TEMPLATES
TEMPORARY	TIME	TIME_FORMAT
TIME_SCALE	time_t	TIME_VALUE
TRANSACTION	TRANSDUCER	TRANSPARENT
TRUE	TUNE	TV
TV1	TV2	TV3
TV4	TV5	TV6
TV7	TV8	TV9
TV10	TV11	TV12
TV13	TV14	TV15
TV16	TV17	TV18

EDDL keyword	EDDL keyword	EDDL keyword
TV19	TV20	TYPE
UNCORRECTABLE	UNIT	UNIT_ITEMS
UNITS	UNIVERSAL	unsigned
UNSIGNED_INTEGER	UUID	
VALIDITY	VARIABLE	VARIABLE_LIST
VARIABLE_LISTS	VARIABLE_STATUS	VARIABLES
VECTORS	VERTICAL	VERTICAL_BAR
VIEW_MAX	VIEW_MIN	VISIBILITY
VISIBLE		
WARNING	WAVEFORM	WAVEFORM_ITEMS
WAVEFORMS	while	WIDTH
WINDOW	WRITE	WRITE_AS_ONE
WRITE_AS_ONE_ITEMS	WRITE_AS_ONES	
X_AXIS	X_INCREMENT	X_INITIAL
X_LARGE	X_SMALL	X_VALUES
XX_LARGE	XX_SMALL	XXX_SMALL
XY		
Y_AXIS	Y_VALUES	YT

A.5 Terminals

The following text contains the allowed terminals of the EDDL lexical structure.

```

DEFINE digit          = { 0-9 } .
    bin_digit         = { 0 1 } .
    non_zero_digit    = { 1-9 '-' } .
    oct_digit         = { 0-7 } .
    hex_digit         = { 0-9abcdefABCDEF } .
    letter            = { a-zA-Z } .
    escapes           = { '\"?afnrtv'\\" } .
    ISOLatin1char     = - { " } .

/* Integer */
(0b|0B) bin_digit +    /* binary */
non_zero_digit digit * /* decimal */
"0" oct_digit *        /* octal */
(0x|0X) hex_digit +    /* hexadecimal */

/* Real */
digit* "." digit + ((E|e){+|-}? digit +)?

/* String */
\" ISOLatin1char * \"
\" UTF8char * \"

/* Character */
\' ISOLatin1char \'
\' UTF8char \'

/* An EDD shall use one and only one character encoding scheme */
/* (i.e., ISO Latin-1 or UTF-8). */

/* Identifier */

```

```
letter (letter|digit|_)*

/* uuid */
/* Universally unique identifier as specified in ISO/IEC 9834-8. */
/* 32 hexadecimal digits in 5 '-' separated groups: 8-4-4-4-12 */
hex_digit hex_digit hex_digit hex_digit hex_digit hex_digit hex_digit
\ hex_digit '-'
\ hex_digit hex_digit hex_digit hex_digit '-'
\ hex_digit hex_digit hex_digit hex_digit '-'
\ hex_digit hex_digit hex_digit hex_digit '-'
\ hex_digit hex_digit hex_digit hex_digit hex_digit hex_digit hex_digit
\ hex_digit hex_digit hex_digit hex_digit hex_digit
```

A.6 Formal EDDL syntax

A.6.1 General

Annex A contains a formal syntax of the EDDL using the Backus Naur Form.

If there are optional elements then each element is marked with /* M */ if the element is mandatory or /* O */ if the element is optional.

Each Backus Naur Form rule is identified by an identifier followed by a colon.

A.6.2 EDD identification information

```
device_description:
    = identification definition_list

identification:
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_profile
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' manufacturer_ext
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' edd_profile
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' manufacturer_ext
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_profile ',' manufacturer_ext
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' edd_profile ','
      manufacturer_ext

manufacturer:
    = 'MANUFACTURER' Integer
    = 'MANUFACTURER' Identifier

device_type:
    = 'DEVICE_TYPE' Integer
    = 'DEVICE_TYPE' Identifier

device_revision:
    = 'DEVICE_REVISION' Integer

DD_revision:
    = 'DD_REVISION' Integer
```

```

    = 'EDD_REVISION' Integer

edd_version:
    = 'EDD_VERSION' Integer

edd_profile:
    = 'EDD_PROFILE' Integer

manufacturer_ext:
    = 'MANUFACTURER_EXT' string

definition_list:
    = definition
    = definition_list definition

definition:
    = item
    = imported_description
    = like

item:
    = axis
    = array
    = blob
    = block
    = chart
    = collection
    = command
    = component
    = component_folder
    = component_reference
    = component_relation
    = edit_display
    = file
    = graph
    = grid
    = list
    = image
    = interface
    = item_array
    = menu
    = method
    = plugin
    = record
    = refresh_relation
    = response_codes_definition
    = source
    = template
    = unit_relation
    = variable
    = variable_list
    = waveform
    = wao_relation

```

A.6.3 AXIS

```

axis:
    = 'AXIS' Identifier '{' axis_attribute_list '}'
    = 'AXIS' Identifier '{' '}'

axis_attribute_list:
    = axis_attribute
    = axis_attribute_list axis_attribute

```

```

axis_attribute:
    = constant_unit                /* O */
    = help                        /* O */
    = label                      /* O */
    = axis_max_value             /* O */
    = axis_min_value            /* O */
    = axis_scaling               /* O */

axis_min_value:
    = minimum_value

axis_max_value:
    = maximum_value

axis_scaling:
    = 'SCALING' axis_scaling_specifier

axis_scaling_specifier:
    = axis_scaling_keyword ';'
    = 'IF' '(' expr ')' '{' axis_scaling_specifier '}'
    = 'IF' '(' expr ')' '{' axis_scaling_specifier '}'
      'ELSE' '{' axis_scaling_specifier '}'
    = 'SELECT' '(' expr ')' '{' axis_scaling_selection_list '}'

axis_scaling_selection_list:
    = axis_scaling_selection
    = axis_scaling_selection_list axis_scaling_selection

axis_scaling_selection:
    = 'CASE' expr ':' axis_scaling_specifier
    = 'DEFAULT' ':' axis_scaling_specifier

axis_scaling_keyword:
    = 'LINEAR'
    = 'LOGARITHMIC'

A.6.4 BLOCK_A and BLOCK_B

block:
    = 'BLOCK' Identifier '{' block_attribute_list '}'

block_attribute_list:
    = block_attribute
    = block_attribute_list block_attribute

block_attribute:
    = block_a_characteristics    /* M */
    = label                    /* M */
    = block_a_parameters        /* M */
    = block_a_axis_items       /* O */
    = block_a_chart_items      /* O */
    = block_a_collection_items /* O */
    = block_a_edit_display_items /* O */
    = block_a_file_items       /* O */
    = block_a_graph_items      /* O */
    = block_a_grid_items       /* O */
    = help                    /* O */
    = block_a_image_items      /* O */
    = block_a_item_array_items /* O */
    = block_a_list_items       /* O */
    = block_a_local_parameters /* O */
    = block_a_menu_items       /* O */
    = block_a_method_items     /* O */
    = block_a_parameter_lists  /* O */

```

```

= block_a_plugin_items          /* O */
= block_a_refresh_items        /* O */
= block_a_source_items         /* O */
= block_a_unit_items           /* O */
= block_a_wao_items            /* O */
= block_a_waveform_items       /* O */
= block_a_charts               /* O */
= block_a_files                /* O */
= block_a_lists                /* O */
= block_a_graphs               /* O */
= block_a_grids                /* O */
= block_a_menus                /* O */
= block_a_methods              /* O */
= block_a_plugins              /* O */
= block_b_number               /* M */
= block_b_type                 /* M */

block_a_characteristics:
= 'CHARACTERISTICS' record_reference ';'

block_a_parameters:
= 'PARAMETERS' '{' member_list '}'

member_list:
= member
= member_list member

block_a_axis_items:
= 'AXIS_ITEMS' '{' axis_items_specifier_list '}'

axis_items_specifier_list:
= axis_items_specifier
= axis_items_specifier_list ',' axis_items_specifier

axis_items_specifier:
= axis_reference

block_a_chart_items:
= 'CHART_ITEMS' '{' chart_items_specifier_list '}'

chart_items_specifier_list:
= chart_items_specifier
= chart_items_specifier_list ',' chart_items_specifier

chart_items_specifier:
= chart_reference

block_a_collection_items:
= 'COLLECTION_ITEMS' '{' collection_items_specifier_list '}'

collection_items_specifier_list:
= collection_items_specifier
= collection_items_specifier_list ',' collection_items_specifier

collection_items_specifier:
= collection_reference

block_a_edit_display_items:
= 'EDIT_DISPLAY_ITEMS' '{' edit_display_items_specifier_list '}'

edit_display_items_specifier_list:
= edit_display_items_specifier
= edit_display_items_specifier_list ',' edit_display_items_specifier

```

```

edit_display_items_specifier:
    = edit_display_reference

block_a_file_items:
    = 'FILE_ITEMS' '{' file_items_specifier_list '}'

file_items_specifier_list:
    = file_items_specifier
    = file_items_specifier_list ',' file_items_specifier

file_items_specifier:
    = file_reference

block_a_graph_items:
    = 'GRAPH_ITEMS' '{' graph_items_specifier_list '}'

graph_items_specifier_list:
    = graph_items_specifier
    = graph_items_specifier_list ',' graph_items_specifier

graph_items_specifier:
    = graph_reference

block_a_grid_items:
    = 'GRID_ITEMS' '{' grid_items_specifier_list '}'

grid_items_specifier_list:
    = grid_items_specifier
    = grid_items_specifier_list ',' grid_items_specifier

grid_items_specifier:
    = grid_reference

block_a_image_items:
    = 'IMAGE_ITEMS' '{' image_items_specifier_list '}'

image_items_specifier_list:
    = image_items_specifier
    = image_items_specifier_list ',' image_items_specifier

image_items_specifier:
    = image_reference

block_a_item_array_items:
    = 'ITEM_ARRAY_ITEMS' '{' item_array_items_specifier_list '}'

item_array_items_specifier_list:
    = item_array_items_specifier
    = item_array_items_specifier_list ',' item_array_items_specifier

item_array_items_specifier:
    = item_array_reference

block_a_list_items:
    = 'LIST_ITEMS' '{' list_items_specifier_list '}'

list_items_specifier_list:
    = list_items_specifier
    = list_items_specifier_list ',' list_items_specifier

list_items_specifier:
    = list_reference

```

```
block_a_local_parameters:
    = 'LOCAL_PARAMETERS' '{' member_list '}'

block_a_menu_items:
    = 'MENU_ITEMS' '{' block_a_menu_items_specifier_list '}'

block_a_menu_items_specifier_list:
    = block_a_menu_items_specifier
    = block_a_menu_items_specifier_list ',' block_a_menu_items_specifier

block_a_menu_items_specifier:
    = menu_reference

block_a_method_items:
    = 'METHOD_ITEMS' '{' method_items_specifier_list '}'

method_items_specifier_list:
    = method_items_specifier
    = method_items_specifier_list ',' method_items_specifier

method_items_specifier:
    = method_reference

block_a_parameter_lists:
    = 'PARAMETER_LISTS' '{' member_list '}'

block_a_plugin_items:
    = 'PLUGIN_ITEMS' '{' plugin_items_specifier_list '}'

plugin_items_specifier_list:
    = plugin_items_specifier
    = plugin_items_specifier_list ',' plugin_items_specifier

plugin_items_specifier:
    = plugin_reference

block_a_refresh_items:
    = 'REFRESH_ITEMS' '{' refresh_items_specifier_list '}'

refresh_items_specifier_list:
    = refresh_items_specifier
    = refresh_items_specifier_list ',' refresh_items_specifier

refresh_items_specifier:
    = refresh_reference

block_a_source_items:
    = 'SOURCE_ITEMS' '{' source_items_specifier_list '}'

source_items_specifier_list:
    = source_items_specifier
    = source_items_specifier_list ',' source_items_specifier

source_items_specifier:
    = source_reference

block_a_unit_items:
    = 'UNIT_ITEMS' '{' unit_items_specifier_list '}'

unit_items_specifier_list:
    = unit_items_specifier
    = unit_items_specifier_list ',' unit_items_specifier
```

```

unit_items_specifier:
    = unit_reference

block_a_wao_items:
    = 'WRITE_AS_ONE_ITEMS' '{' wao_items_specifier_list '}'

wao_items_specifier_list:
    = wao_items_specifier
    = wao_items_specifier_list ',' wao_items_specifier

wao_items_specifier:
    = wao_reference

block_a_waveform_items:
    = 'WAVEFORM_ITEMS' '{' waveform_items_specifier_list '}'

waveform_items_specifier_list:
    = waveform_items_specifier
    = waveform_items_specifier_list ',' waveform_items_specifier

waveform_items_specifier:
    = waveform_reference

block_a_charts:
    = 'CHARTS' '{' member_list '}'

block_a_files:
    = 'FILES' '{' member_list '}'

block_a_lists:
    = 'LISTS' '{' member_list '}'

block_a_graphs:
    = 'GRAPHS' '{' member_list '}'

block_a_grids:
    = 'GRIDS' '{' member_list '}'

block_a_menus:
    = 'MENUS' '{' member_list '}'

block_a_methods:
    = 'METHODS' '{' member_list '}'

block_a_plugins:
    = 'PLUGINS' '{' member_list '}'

block_b_number:
    = 'NUMBER' expr ';'

block_b_type:
    = 'TYPE' 'PHYSICAL' ';'
    = 'TYPE' 'TRANSDUCER' ';'
    = 'TYPE' 'FUNCTION' ';'

```

A.6.5 CHART

```

chart:
    = 'CHART' Identifier '{' chart_attribute_list '}'

chart_attribute_list:
    = chart_attribute
    = chart_attribute_list chart_attribute

```



```

chart_attribute:
    = members                      /* M */
    = chart_cycle_time             /* O */
    = help                        /* O */
    = height                      /* O */
    = label                       /* O */
    = chart_length                 /* O */
    = chart_type                  /* O */
    = validity                    /* O */
    = visibility                  /* O */
    = width                      /* O */

chart_cycle_time:
    = 'CYCLE_TIME' expr_specifier

chart_length:
    = 'LENGTH' expr_specifier

chart_type:
    = 'TYPE' chart_type_specifier

chart_type_specifier:
    = chart_type_keyword ';'
    = 'IF' '(' expr ')' '{' chart_type_specifier '}'
    = 'IF' '(' expr ')' '{' chart_type_specifier '}'
    'ELSE' '{' chart_type_specifier '}'
    = 'SELECT' '(' expr ')' '{' chart_type_selection_list '}'

chart_type_selection_list:
    = chart_type_selection
    = chart_type_selection_list chart_type_selection

chart_type_selection:
    = 'CASE' expr ':' chart_type_specifier
    = 'DEFAULT' ':' chart_type_specifier

chart_type_keyword:
    = 'GAUGE'
    = 'HORIZONTAL_BAR'
    = 'SCOPE'
    = 'STRIP'
    = 'SWEEP'
    = 'VERTICAL_BAR'

```

A.6.6 COLLECTION

```

collection:
    = 'COLLECTION' Identifier '{' collection_attribute_list '}'
    = 'COLLECTION' 'OF' collection_item_type Identifier '{'
    collection_attribute_list '}'

collection_item_type:
    = 'ARRAY'
    = 'BLOCK'
    = 'CHART'
    = 'COLLECTION' 'OF' collection_item_type
    = 'EDIT_DISPLAY'
    = 'FILE'
    = 'GRAPH'
    = 'GRID'
    = 'IMAGE'
    = 'ITEM_ARRAY' 'OF' collection_item_type
    = 'LIST'
    = 'MENU'

```

```
= 'METHOD'
= 'PLUGIN'
= 'RECORD'
= 'VARIABLE'
= 'VARIABLE_LIST'
```

```
item_type:
= collection_item_type
= 'AXIS'
= 'COMMAND'
= 'COMPONENT'
= 'COMPONENT_FOLDER'
= 'COMPONENT_REFERENCE'
= 'COMPONENT_RELATION'
= 'INTERFACE'
= 'REFRESH'
= 'RESPONSE_CODES'
= 'SOURCE'
= 'TEMPLATE'
= 'UNIT'
= 'WAVEFORM'
= 'WRITE_AS_ONE'
```

```
collection_attribute_list:
= collection_attribute
= collection_attribute_list collection_attribute
```

```
collection_attribute:
= members /* M */
= help /* O */
= label /* O */
= private /* O */
= validity /* O */
= visibility /* O */
```

A.6.7 COMMAND

```
command:
= 'COMMAND' Identifier '{' command_attribute_list '}'
= 'COMMAND' Identifier '{' '}'
```

```
command_attribute_list:
= command_attribute
= command_attribute_list command_attribute
```

```
command_attribute:
= command_operation /* M */
= command_transaction /* M */
= command_index /* O */
= command_block /* O */
= command_number /* O */
= command_slot /* O */
= command_sub_slot /* O */
= command_api /* O */
= command_header /* O */
= post_rqstreceive_actions /* O */
= response_codes /* O */
```

```
command_operation:
= 'OPERATION' operation ';' ;'
```

```
operation:
= 'READ'
= 'WRITE'
```

```

= 'COMMAND'

command_transaction:
= 'TRANSACTION' '{' transaction_attribute_list '}'
= 'TRANSACTION' Integer '{' transaction_attribute_list '}'

transaction_attribute_list:
= transaction_attribute
= transaction_attribute_list transaction_attribute

transaction_attribute:
= request
= reply
= 'RESPONSE_CODES' '(' response_code_reference ')'

request:
= 'REQUEST' '{' data_items_specifier_list '}'
= 'REQUEST' '{' '}'

reply:
= 'REPLY' '{' data_items_specifier_list '}'
= 'REPLY' '{' '}'

data_items_specifier_list:
= data_items_specifier
= data_items_specifier_list ',' data_items_specifier

data_items_specifier:
= Integer
= variable_reference
= variable_reference '<' Integer '>'
= variable_reference '(' data_items_qualifier_list ')'
= variable_reference '<' Integer '>' '(' data_items_qualifier_list ')'
= array_reference
= array_reference '(' data_items_qualifier_list ')'
= collection_reference
= collection_reference '(' data_items_qualifier_list ')'
= item_array_reference
= item_array_reference '(' data_items_qualifier_list ')'
= list_reference
= list_reference '(' data_items_qualifier_list ')'

data_items_qualifier_list:
= data_items_qualifier
= data_items_qualifier_list ',' data_items_qualifier

data_items_qualifier:
= 'INDEX'
= 'INFO'

command_index:
= 'INDEX' index_specifier

index_specifier:
= index_items ';'
= 'IF' '(' expr ')' '{' index_specifier '}'
= 'IF' '(' expr ')' '{' index_specifier '}'
= 'ELSE' '{' index_specifier '}'
= 'SELECT' '(' expr ')' '{' index_selection_list '}'

index_items:
= expr
= variable_reference

```

```

index_selection_list:
    = index_selection
    = index_selection_list index_selection

index_selection:
    = 'CASE' expr ':' index_specifier
    = 'DEFAULT' ':' index_specifier

command_block:
    = 'BLOCK' block_specifier

block_specifier:
    = block_b_reference ';'
    = 'IF' '(' expr ')' '{' block_specifier '}'
    = 'IF' '(' expr ')' '{' block_specifier '}'
    'ELSE' '{' block_specifier '}'
    = 'SELECT' '(' expr ')' '{' block_selection_list '}'

block_selection_list:
    = block_selection
    = block_selection_list block_selection

block_selection:
    = 'CASE' expr ':' block_specifier
    = 'DEFAULT' ':' block_specifier

command_number:
    = 'NUMBER' Integer ';'

command_slot:
    = 'SLOT' slot_specifier

slot_specifier:
    = slot_items ';'
    = 'IF' '(' expr ')' '{' slot_specifier '}'
    = 'IF' '(' expr ')' '{' slot_specifier '}'
    'ELSE' '{' slot_specifier '}'
    = 'SELECT' '(' expr ')' '{' slot_selection_list '}'

slot_items:
    = expr
    = variable_reference

slot_selection_list:
    = slot_selection
    = slot_selection_list slot_selection

slot_selection:
    = 'CASE' expr ':' slot_specifier
    = 'DEFAULT' ':' slot_specifier

command_sub_slot:
    = 'SUB_SLOT' slot_specifier

command_api:
    = 'API' api_specifier

api_specifier:
    = expr ';'
    = 'IF' '(' expr ')' '{' api_specifier '}'
    = 'IF' '(' expr ')' '{' api_specifier '}' 'ELSE' '{' api_specifier '}'
    = 'SELECT' '(' expr ')' '{' api_selection_list '}'

```

```

api_selection_list:
    = api_selection
    = api_selection_list api_selection

api_selection:
    = 'CASE' expr ':' api_specifier
    = 'DEFAULT' ':' api_specifier

command_header:
    = 'HEADER' header_specifier

header_specifier:
    = string ';'
    = 'IF' '(' expr ')' '{' header_specifier '}'
    = 'IF' '(' expr ')' '{' header_specifier '}'
    = 'ELSE' '{' header_specifier '}'
    = 'SELECT' '(' expr ')' '{' header_selection_list '}'

header_selection_list:
    = header_selection
    = header_selection_list header_selection

header_selection:
    = 'CASE' expr ':' header_specifier
    = 'DEFAULT' ':' header_specifier

post_rqstreceive_actions:
    = 'POST_RQSTRECEIVE_ACTIONS' '{' actions_specifier_list '}'

```

A.6.8 COMPONENT

```

component:
    = 'COMPONENT' Identifier '{' component_attribute_list '}'

component_attribute_list:
    = component_attribute
    = component_attribute_list component_attribute

component_attribute:
    = label /* M */
    = byte_order /* O */
    = can_delete /* O */
    = check_configuration /* O */
    = classification /* O */
    = component_parent /* O */
    = component_path /* O */
    = component_relations /* O */
    = connection_point /* O */
    = declaration /* O */
    = detect_method /* O */
    = edd /* O */
    = help /* O */
    = initial_values /* O */
    = redundancy /* O */
    = product_uri /* O */
    = protocol /* O */
    = scan_method /* O */
    = scan_list /* O */
    = supplied_interface /* O */

byte_order:
    = 'BYTE_ORDER' byte_order_specifier ';'

```

```
byte_order_specifier:
    = 'BIG_ENDIAN'
    = 'LITTLE_ENDIAN'

can_delete:
    = 'CAN_DELETE' boolean_specifier

check_configuration:
    = 'CHECK_CONFIGURATION' method_reference ';'

classification:
    = 'CLASSIFICATION' classification_specifier ';'

classification_specifier:
    = 'ACTUATOR'
    = 'ACTUATOR_ELECTRO_PNEUMATIC'
    = 'ACTUATOR_ELECTRIC'
    = 'ACTUATOR_HYDRAULIC'
    = 'CONVERTER'
    = 'CONTROLLER'
    = 'DISCRETE_IN'
    = 'DISCRETE_OUT'
    = 'ELECTRICAL_DISTRIBUTION'
    = 'ELECTRICAL_DISTRIBUTION_POWER_MONITORING'
    = 'FREQUENCY_CONVERTER'
    = 'INDICATOR'
    = 'NETWORK'
    = 'NETWORK_COMMUNICATION_SERVICE_PROVIDER'
    = 'NETWORK_COMPONENT'
    = 'NETWORK_COMPONENT_WIRELESSADAPTER'
    = 'NETWORK_CONNECTION_POINT'
    = 'REMOTEIO'
    = 'RECORDER'
    = 'SENSOR'
    = 'SENSOR_ACOUSTIC'
    = 'SENSOR_ANALYTIC'
    = 'SENSOR_ANALYTIC_GAS'
    = 'SENSOR_ANALYTIC_LIQUID'
    = 'SENSOR_CONCENTRATION'
    = 'SENSOR_DENSITY'
    = 'SENSOR_DENSITY_RADIOMETRIC'
    = 'SENSOR_FLOW'
    = 'SENSOR_FLOW_CORIOLIS'
    = 'SENSOR_FLOW_ELECTRO_MAGNETIC'
    = 'SENSOR_FLOW_MECHANICAL'
    = 'SENSOR_FLOW_RADIOMETRIC'
    = 'SENSOR_FLOW_THERMAL'
    = 'SENSOR_FLOW_ULTRASONIC'
    = 'SENSOR_FLOW_VORTEX_COUNTER'
    = 'SENSOR_LEVEL'
    = 'SENSOR_LEVEL_BUOYANCY'
    = 'SENSOR_LEVEL_CAPACITIVE'
    = 'SENSOR_LEVEL_ECHO'
    = 'SENSOR_LEVEL_HYDROSTATIC'
    = 'SENSOR_LEVEL_RADIOMETRIC'
    = 'SENSOR_MULTIVARIABLE'
    = 'SENSOR_POSITION'
    = 'SENSOR_PRESSURE'
    = 'SENSOR_TEMPERATURE'
    = 'SEMICONDUCTOR'
    = 'SWITCHGEAR'
    = 'UNIVERSAL'
```

```
= string

component_path:
    = 'COMPONENT_PATH' string ';'

component_parent:
    = 'COMPONENT_PARENT' component_folder_reference ';'

component_relations:
    = 'COMPONENT_RELATIONS' '{' component_relations_specifier_list '}'

component_relations_specifier_list:
    = component_relations_specifier
    = component_relations_specifier_list ',' component_relations_specifier

component_relations_specifier:
    = component_relation_reference
    = 'IF' '(' expr ')' '{' component_relations_specifier_list '}'
    = 'IF' '(' expr ')' '{' component_relations_specifier_list '}'
    'ELSE' '{' component_relations_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' component_relations_selection_list '}'

component_relations_selection_list:
    = component_relations_selection
    = component_relations_selection_list component_relations_selection

component_relations_selection:
    = 'CASE' expr ':' component_relations_specifier_list
    = 'DEFAULT' ':' component_relations_specifier_list

connection_point:
    = 'CONNECTION_POINT' collection_reference ';'

declaration:
    = 'DECLARATION' '{' item_list '}'

item_list:
    = item
    = item_list item

detect_method:
    = 'DETECT' method_reference ';'

edd:
    = 'EDD' string ';'

product_uri:
    = 'PRODUCT_URI' string_literal ';'

protocol:
    = 'PROTOCOL' protocol_specifier ';'

protocol_specifier:
    = 'PROFIBUS_DP'
    = 'PROFIBUS_PA'
    = 'PROFINET'
    = 'HART'
    = 'FF'
    = string

redundancy:
    = 'REDUNDANCY' expr_specifier
```

```

initial_values:
    = 'INITIAL_VALUES' '{' initial_value_list '}'

initial_value_list:
    = initial_value_specifier
    = initial_value_list initial_value_specifier

initial_value_specifier:
    = reference_without_call '=' expr_specifier
    = reference_without_call '=' array_initialization

array_initialization:
    = '{' array_initialization_list '}'
    = '{' array_initialization_element_list '}'

array_initialization_list:
    = array_initialization
    = array_initialization_list ',' array_initialization

array_initialization_element_list:
    = array_initialization_element
    = array_initialization_list ',' array_initialization_element

array_initialization_element:
    = conditional_expr

scan_method:
    = 'SCAN' method_reference ';'

scan_list:
    = 'SCAN_LIST' list_reference ';'

supplied_interface:
    = 'SUPPLIED_INTERFACE' '{' interface_list '}'

```

A.6.9 COMPONENT_FOLDER

```

component_folder:
    = 'COMPONENT_FOLDER' Identifier '{' component_folder_attribute_list '}'

component_folder_attribute_list:
    = component_folder_attribute
    = component_folder_attribute_list component_folder_attribute

component_folder_attribute:
    = label /* M */
    = classification /* O */
    = component_parent /* O */
    = component_path /* O */
    = help /* O */
    = protocol /* O */

```

A.6.10 COMPONENT_REFERENCE

```

component_reference:
    = 'COMPONENT_REFERENCE' Identifier '{'
        component_reference_attribute_comp_list '}'
    = 'COMPONENT_REFERENCE' Identifier '{'
        component_reference_attribute_edd_list '}'

component_reference_attribute_comp_list:
    = component_reference_attribute_comp
    = component_reference_attribute_comp_list
        component_reference_attribute_comp

```



```

component_reference_attribute_edd_list:
    = component_reference_attribute_edd
    = component_reference_attribute_edd_list
    component_reference_attribute_edd

```

```

component_reference_attribute_comp:
    = protocol /* O */
    = classification /* O */
    = component_parent /* O */
    = component_path /* O */

```

```

component_reference_attribute_edd:
    = device_revision_attribute /* O */
    = device_type_attribute /* O */
    = manufacturer_attribute /* O */

```

```

device_revision_attribute:
    = device_revision ';'

```

```

device_type_attribute:
    = device_type ';'

```

```

manufacturer_attribute:
    = manufacturer ';'

```

A.6.11 COMPONENT_RELATION

```

component_relation:
    = 'COMPONENT_RELATION' Identifier '{'
    component_relation_attribute_list '}'

```

```

component_relation_attribute_list:
    = component_relation_attribute
    = component_relation_attribute_list component_relation_attribute

```

```

component_relation_attribute:
    = components /* M */
    = label /* M */
    = relation_type /* M */
    = addressing /* O */
    = help /* O */
    = maximum_number /* O */
    = minimum_number /* O */
    = required_interface /* O */
    = supplied_interface /* O */

```

```

components:
    = 'COMPONENTS' '{' components_specifier_list '}'

```

```

components_specifier_list:
    = components_specifier
    = components_specifier_list ',' components_specifier

```

```

components_specifier:
    = components_reference
    = 'IF' '(' expr ')' '{' components_specifier_list '}'
    = 'IF' '(' expr ')' '{' components_specifier_list '}'
    'ELSE' '{' components_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' components_selection_list '}'

```

```

components_selection_list:
    = components_selection
    = components_selection_list components_selection

```

```

components_selection:
    = 'CASE' expr ':' components_specifier_list
    = 'DEFAULT' ':' components_specifier_list

components_reference:
/* reference to COMPONENT_FOLDER or COMPONENT */
    = reference_without_call
    = reference_without_call '{' component_arg_list '}'

component_arg_list:
    = component_arg
    = component_arg_list component_arg

component_arg:
    = minimum_number
    = maximum_number
    = auto_create
    = filter
    = required_ranges

auto_create:
    = 'AUTO_CREATE' expr_specifier

filter:
    = 'FILTER' '(' expr ')'

required_ranges:
    = 'REQUIRED_RANGES' '{' required_ranges_list '}'

required_ranges_list:
    = required_range_specifier
    = required_ranges_list ';' required_range_specifier

required_range_specifier:
    = variable_reference '{' variable_range_arg_list '}'

variable_range_arg_list:
    = variable_range_arg
    = variable_range_arg_list variable_range_arg

variable_range_arg:
    = maximum_value
    = maximum_value_n
    = minimum_value
    = minimum_value_n

relation_type:
    = 'RELATION_TYPE' relation_type_specifier ';'

relation_type_specifier:
    = 'PARENT_COMPONENT'
    = 'CHILD_COMPONENT'
    = 'SIBLING_COMPONENT'
    = 'NEXT_COMPONENT'
    = 'PREV_COMPONENT'

addressing:
    = 'ADDRESSING' '{' addressing_list '}'

addressing_list:
    = variable_reference
    = addressing_list ',' variable_reference

```

```

maximum_number:
    = 'MAXIMUM_NUMBER' expr_specifier

minimum_number:
    = 'MINIMUM_NUMBER' expr_specifier

required_interface:
    = 'REQUIRED_INTERFACE' '{' interface_list '}'

interface_list:
    = interface_reference
    = interface_list ',' interface_reference

```

A.6.12 CONNECTION (void)

A.6.13 DOMAIN (void)

A.6.14 EDIT_DISPLAY

```

edit_display:
    = 'EDIT_DISPLAY' Identifier '{' edit_display_attribute_list '}'

edit_display_attribute_list:
    = edit_display_attribute
    = edit_display_attribute_list edit_display_attribute

edit_display_attribute:
    = edit_items /* M */
    = label /* M */
    = display_items /* O */
    = post_edit_actions /* O */
    = pre_edit_actions /* O */

edit_items:
    = 'EDIT_ITEMS' '{' edit_items_specifier_list '}'

edit_items_specifier_list:
    = edit_items_specifier
    = edit_items_specifier_list edit_items_specifier

edit_items_specifier:
    = edit_item_list
    = 'IF' '(' expr ')' '{' edit_items_specifier_list '}'
    = 'IF' '(' expr ')' '{' edit_items_specifier_list '}'
    = 'ELSE' '{' edit_items_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' edit_items_selection_list '}'

edit_items_selection_list:
    = edit_items_selection
    = edit_items_selection_list edit_items_selection

edit_items_selection:
    = 'CASE' expr ':' edit_items_specifier_list
    = 'DEFAULT' ':' edit_items_specifier_list

edit_item_list:
    = edit_item
    = edit_item_list ',' edit_item

edit_item:
    = reference_without_call

display_items:
    = 'DISPLAY_ITEMS' '{' display_items_specifier_list '}'

```

```

display_items_specifier_list:
    = display_items_specifier
    = display_items_specifier_list display_items_specifier

display_items_specifier:
    = display_item_list
    = 'IF' '(' expr ')' '{' display_items_specifier_list '}'
    = 'IF' '(' expr ')' '{' display_items_specifier_list '}'
    = 'ELSE' '{' display_items_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' display_items_selection_list '}'

display_items_selection_list:
    = display_items_selection
    = display_items_selection_list display_items_selection

display_items_selection:
    = 'CASE' expr ':' display_items_specifier_list
    = 'DEFAULT' ':' display_items_specifier_list

display_item_list:
    = display_item
    = display_item_list ',' display_item

display_item:
    = variable_reference

```

A.6.15 FILE

```

file:
    = 'FILE' Identifier '{' file_attribute_list '}'

file_attribute_list:
    = file_attribute
    = file_attribute_list file_attribute

file_attribute:
    = members /* M */
    = label /* O */
    = help /* O */
    = handling /* O */
    = identity /* O */
    = file_shared /* O */
    = on_update_actions /* O */

file_shared:
    = 'SHARED' boolean ';'

on_update_actions:
    = 'ON_UPDATE_ACTIONS' '{' actions_specifier_list '}'

```

A.6.16 GRAPH

```

graph:
    = 'GRAPH' Identifier '{' graph_attribute_list '}'

graph_attribute_list:
    = graph_attribute
    = graph_attribute_list graph_attribute

graph_attribute:
    = members /* M */
    = label /* O */
    = help /* O */

```

```

= height                /* O */
= graph_cycle_time      /* O */
= validity              /* O */
= visibility            /* O */
= width                 /* O */
= graph_x_axis          /* O */

```

```

graph_cycle_time:
= 'CYCLE_TIME' expr_specifier

```

```

graph_x_axis:
= 'X_AXIS' axis_specifier

```

A.6.17 GRID

```

grid:
= 'GRID' Identifier '{' grid_attribute_list '}'

```

```

grid_attribute_list:
= grid_attribute
= grid_attribute_list grid_attribute

```

```

grid_attribute:
= grid_vectors          /* M */
= handling              /* O */
= height                /* O */
= help                  /* O */
= label                 /* O */
= grid_orientation      /* O */
= validity              /* O */
= visibility            /* O */
= width                 /* O */

```

```

grid_vectors:
= 'VECTORS' '{' vector_specifier_list '}'

```

```

vector_specifier_list:
= vector_specifier
= vector_specifier_list ',' vector_specifier

```

```

vector_specifier:
= vector
= 'IF' '(' expr ')' '{' vector_specifier_list '}'
= 'IF' '(' expr ')' '{' vector_specifier_list '}'
'ELSE' '{' vector_specifier_list '}'
= 'SELECT' '(' expr ')' '{' vector_selection_list '}'

```

```

vector_selection_list:
= vector_selection
= vector_selection_list vector_selection

```

```

vector_selection:
= 'CASE' expr ':' vector_specifier_list
= 'DEFAULT' ':' vector_specifier_list

```

```

vector:
= '{' string ',' vector_values_list '}'

```

```

vector_values_list:
= vector_values
= vector_values_list ',' vector_values

```

```

vector_values:
= reference_without_call

```

= Integer
= Real
= string

grid_orientation:
= 'ORIENTATION' orientation_specifier ';'

orientation_specifier:
= 'HORIZONTAL'
= 'VERTICAL'

A.6.18 IMAGE

image:
= 'IMAGE' Identifier '{' image_attribute_list '}'

image_attribute_list:
= image_attribute
= image_attribute_list image_attribute

image_attribute:
= image_path
= label
= help
= image_link
= validity
= visibility

```
/* M */
/* O */
/* O */
/* O */
/* O */
/* O */
/* O */
```

image_path:
= 'PATH' file_name_specifier

file_name_specifier:
= string_literal ';'
= 'IF' '(' expr ')' '{' file_name_specifier '}'
= 'IF' '(' expr ')' '{' file_name_specifier '}'
= 'ELSE' '{' file_name_specifier '}'
= 'SELECT' '(' expr ')' '{' file_name_selection_list '}'

file_name_selection_list:
= file_name_selection
= file_name_selection_list file_name_selection

file_name_selection:
= 'CASE' expr ':' file_name_specifier
= 'DEFAULT' ':' file_name_specifier

image_link:
= 'LINK' link_specifier

link_specifier:
= reference_without_call ';'
= 'IF' '(' expr ')' '{' link_specifier '}'
= 'IF' '(' expr ')' '{' link_specifier '}'
= 'ELSE' '{' link_specifier '}'
= 'SELECT' '(' expr ')' '{' link_selection_list '}'

link_selection_list:
= link_selection
= link_selection_list link_selection

link_selection:
= 'CASE' expr ':' link_specifier
= 'DEFAULT' ':' link_specifier

A.6.19 INTERFACE

```

interface:
    = 'INTERFACE' Identifier '{' interface_attribute_list '}'

interface_attribute_list:
    = interface_attribute
    = interface_attribute_list interface_attribute

interface_attribute:
    = declaration                /* M */
    = label                      /* M */
    = help                      /* O */

```

A.6.20 LIST

```

list:
    = 'LIST' Identifier '{' list_attribute_list '}'

list_attribute_list:
    = list_attribute
    = list_attribute_list list_attribute

list_attribute:
    = list_type                /* M */
    = list_capacity            /* O */
    = list_count               /* O */
    = help                    /* O */
    = label                   /* O */
    = private                 /* O */
    = validity                /* O */
    = visibility              /* O */

```

```

list_type:
    = 'TYPE' list_type_specifier ';'

```

```

list_type_specifier:
    = reference_without_call

```

```

list_capacity:
    = 'CAPACITY' Integer ';'

```

```

list_count:
    = 'COUNT' expr_specifier

```

A.6.21 IMPORT

```

imported_description:
    = 'IMPORT' import_identification '{' imports '}'
    = 'IMPORT' import_identification '{' imports redefinitions '}'
    = 'IMPORT' '"' filename '"'
    = 'IMPORT' '<' filename '>'

```

```

import_identification:
    = import_manufacturer import_device_type ',' device_revision ','
      DD_revision
    = import_manufacturer import_device_type ',' device_revision ','
      DD_revision ',' edd_version
    = import_manufacturer import_device_type ',' device_revision ','
      DD_revision ',' edd_profile
    = import_manufacturer import_device_type ',' device_revision ','
      DD_revision ',' manufacturer_ext
    = import_manufacturer import_device_type ',' device_revision ','
      DD_revision ',' edd_version ',' edd_profile
    = import_manufacturer import_device_type ',' device_revision ','

```

```

    DD_revision ',' edd_version ',' manufacturer_ext
= import_manufacturer import_device_type ',' device_revision ','
    DD_revision ',' edd_profile ',' manufacturer_ext
= import_manufacturer import_device_type ',' device_revision ','
    DD_revision ',' edd_version ',' edd_profile ','
    manufacturer_ext

import_manufacturer:
    = manufacturer ','
    = Identifier

import_device_type:
    = device_type
    = Identifier

imports:
    = 'EVERYTHING' ';'
    = item_import_list

item_import_list:
    = item_import
    = item_import_list item_import

item_import:
    = item_import_by_name ';'
    = item_import_by_type ';'

item_import_by_name:
    = item_type Identifier
    = Identifier

item_import_by_type:
    = import_item_type
    = item_import_by_type '&' import_item_type

import_item_type:
    = 'ARRAYS'
    = 'AXES'
    = 'BLOCKS'
    = 'CHARTS'
    = 'COLLECTIONS'
    = 'COMMANDS'
    = 'COMPONENTS'
    = 'COMPONENT_FOLDERS'
    = 'COMPONENT_REFERENCES'
    = 'COMPONENT_RELATIONS'
    = 'EDIT_DISPLAYS'
    = 'FILES'
    = 'GRAPHS'
    = 'GRIDS'
    = 'IMAGES'
    = 'INTERFACES'
    = 'ITEM_ARRAYS'
    = 'LISTS'
    = 'MENUS'
    = 'METHODS'
    = 'PLUGINS'
    = 'RECORDS'
    = 'REFRESHES'
    = 'RELATIONS'
    = 'RESPONSE_CODES'
    = 'SOURCES'
    = 'TEMPLATES'

```



```

= 'UNITS'
= 'VARIABLES'
= 'VARIABLE_LISTS'
= 'WRITE_AS_ONES'
= 'WAVEFORMS'

```

```

redefinitions:
= 'REDEFINITIONS' '{' redefinition_list '}'

```

```

redefinition_list:
= redefinition
= redefinition_list redefinition

```

```

redefinition:
= array_redefinition
= axis_redefinition
= blob_redefinition
= block_redefinition
= chart_redefinition
= collection_redefinition
= command_redefinition
= component_redefinition
= component_folder_redefinition
= component_reference_redefinition
= component_relation_redefinition
= edit_display_redefinition
= file_redefinition
= graph_redefinition
= grid_redefinition
= image_redefinition
= interface_redefinition
= item_array_redefinition
= list_redefinition
= menu_redefinition
= method_redefinition
= plugin_redefinition
= record_redefinition
= refresh_relation_redefinition
= response_codes_definition_redefinition
= source_redefinition
= template_redefinition
= unit_relation_redefinition
= variable_redefinition
= variable_list_redefinition
= wao_relation_redefinition
= waveform_redefinition

```

```

filename:
= string_literal

```

A.6.22 LIKE

```

like:
= Identifier 'LIKE' Identifier
  '{' attribute_redefinition_list '}'
= Identifier 'LIKE' 'ARRAY' Identifier
  '{' array_attribute_redefinition_list '}'
= Identifier 'LIKE' 'AXIS' Identifier
  '{' axis_attribute_redefinition_list '}'
= Identifier 'LIKE' 'BLOB' Identifier
  '{' blob_attribute_redefinition_list '}'
= Identifier 'LIKE' 'BLOCK' Identifier
  '{' block_attribute_redefinition_list '}'
= Identifier 'LIKE' 'CHART' Identifier

```

```
'{' chart_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COLLECTION' 'OF' collection_item_type Identifier
'{' collection_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COMMAND' Identifier
'{' command_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COMPONENT' Identifier
'{' component_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COMPONENT_FOLDER' Identifier
'{' component_folder_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COMPONENT_REFERENCE' Identifier
'{' component_reference_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COMPONENT_RELATION' Identifier
'{' component_relation_attribute_redefinition_list '}'
= Identifier 'LIKE' 'EDIT_DISPLAY' Identifier
'{' edit_display_attribute_redefinition_list '}'
= Identifier 'LIKE' 'FILE' Identifier
'{' file_attribute_redefinition_list '}'
= Identifier 'LIKE' 'GRAPH' Identifier
'{' graph_attribute_redefinition_list '}'
= Identifier 'LIKE' 'GRID' Identifier
'{' grid_attribute_redefinition_list '}'
= Identifier 'LIKE' 'IMAGE' Identifier
'{' image_attribute_redefinition_list '}'
= Identifier 'LIKE' 'INTERFACE' Identifier
'{' interface_attribute_redefinition_list '}'
= Identifier 'LIKE' 'ITEM_ARRAY' 'OF' collection_item_type
Identifier
'{' item_array_attribute_redefinition_list '}'
= Identifier 'LIKE' 'LIST' Identifier
'{' list_attribute_redefinition_list '}'
= Identifier 'LIKE' 'MENU' Identifier
'{' menu_attribute_redefinition_list '}'
= Identifier 'LIKE' 'METHOD' Identifier
'{' method_attribute_redefinition_list '}'
= Identifier 'LIKE' 'PLUGIN' Identifier
'{' plugin_attribute_redefinition_list '}'
= Identifier 'LIKE' 'RECORD' Identifier
'{' record_attribute_redefinition_list '}'
= Identifier 'LIKE' 'REFRESH' Identifier
'{' refresh_relation_left ':' refresh_relation_right '}'
= Identifier 'LIKE' 'RESPONSE_CODES' Identifier
'{' response_code_redefinition_list '}'
= Identifier 'LIKE' 'SOURCE' Identifier
'{' source_attribute_redefinition_list '}'
= Identifier 'LIKE' 'TEMPLATE' Identifier
'{' template_attribute_redefinition_list '}'
= Identifier 'LIKE' 'UNIT' Identifier
'{' unit_relation_left ':' unit_relation_right '}'
= Identifier 'LIKE' 'VARIABLE' Identifier
'{' variable_attribute_redefinition_list '}'
= Identifier 'LIKE' 'VARIABLE_LIST' Identifier
'{' variable_list_attribute_redefinition_list '}'
= Identifier 'LIKE' 'WAVEFORM' Identifier
'{' waveform_attribute_redefinition_list '}'
= Identifier 'LIKE' 'WRITE_AS_ONE' Identifier
'{' wao_specifier '}'
```

```
attribute_redefinition_list:
    = attribute_redefinition
    = attribute_redefinition_list attribute_redefinition
```

```
attribute_redefinition:
    = array_attribute_redefinition
```

```

= axis_attribute_redefinition
= blob_attribute_redefinition
= block_attribute_redefinition
= chart_attribute_redefinition
= collection_attribute_redefinition
= command_attribute_redefinition
= component_attribute_redefinition
= component_folder_attribute_redefinition
= component_reference_attribute_redefinition
= component_relation_attribute_redefinition
= edit_display_attribute_redefinition
= file_attribute_redefinition
= graph_attribute_redefinition
= grid_attribute_redefinition
= image_attribute_redefinition
= interface_attribute_redefinition
= item_array_attribute_redefinition
= list_attribute_redefinition
= menu_attribute_redefinition
= method_attribute_redefinition
= plugin_attribute_redefinition
= record_attribute_redefinition
= source_attribute_redefinition
= template_attribute_redefinition
= variable_attribute_redefinition
= variable_list_attribute_redefinition
= waveform_attribute_redefinition

```

A.6.23 MENU

```

menu:
    = 'MENU' Identifier '{' menu_attribute_list '}'

```

```

menu_attribute_list:
    = menu_attribute
    = menu_attribute_list menu_attribute

```

```

menu_attribute:
    = menu_items                /* M */
    = label                     /* M */
    = menu_access               /* O */
    = help                     /* O */
    = menu_exit_actions        /* O */
    = menu_init_actions        /* O */
    = post_edit_actions        /* O */
    = post_read_actions        /* O */
    = post_write_actions       /* O */
    = pre_edit_actions         /* O */
    = pre_read_actions         /* O */
    = pre_write_actions        /* O */
    = menu_style               /* O */
    = validity                 /* O */
    = visibility               /* O */

```

```

menu_items:
    = 'ITEMS' '{' '}'
    = 'ITEMS' '{' menu_item_specifier_list '}'

```

```

menu_item_specifier_list:
    = menu_item_specifier
    = menu_item_specifier_list menu_item_specifier

```

```

menu_item_specifier:
    = menu_item_list

```

```

= 'IF' '(' expr ')' '{' menu_item_specifier_list '}'
= 'IF' '(' expr ')' '{' menu_item_specifier_list '}'
  'ELSE' '{' menu_item_specifier_list '}'
= 'SELECT' '(' expr ')' '{' menu_item_selection_list '}'

menu_item_selection_list:
= menu_item_selection
= menu_item_selection_list menu_item_selection

menu_item_selection:
= 'CASE' expr ':' menu_item_specifier_list
= 'DEFAULT' ':' menu_item_specifier_list

menu_item_list:
= menu_item
= menu_item_list ',' menu_item

menu_item:
= reference_without_call
= reference_without_call menu_item_args
= 'SEPARATOR'
= 'COLUMNBREAK'
= 'ROWBREAK'

menu_item_args:
= '(' argument_list ')'
= '(' 'REVIEW' ')'
= '(' variable_qualifier_list ')'
= '(' image_qualifier_list ')'

variable_qualifier_list:
= variable_qualifier
= variable_qualifier_list ',' variable_qualifier

variable_qualifier:
= 'DISPLAY_VALUE'
= 'READ_ONLY'
= 'HIDDEN'
= 'NO_LABEL'
= 'NO_UNIT'

image_qualifier_list:
= image_qualifier
= image_qualifier_list ',' image_qualifier

image_qualifier:
= 'ALIGN_LEFT'
= 'ALIGN_RIGHT'
= 'INLINE'

menu_access:
= 'ACCESS' 'ONLINE' ';'
= 'ACCESS' 'OFFLINE' ';'

menu_style:
= 'STYLE' menu_style_item ';'

menu_style_item:
= 'DIALOG'
= 'GROUP'
= 'MENU'
= 'PAGE'
= 'TABLE'

```

```
= 'WINDOW'
```

```
menu_exit_actions:
    = 'EXIT_ACTIONS' '{' actions_specifier_list '}'
```

```
menu_init_actions:
    = 'INIT_ACTIONS' '{' actions_specifier_list '}'
```

A.6.24 METHOD

```
method:
    = 'METHOD' Identifier '{' method_attribute_list '}'
    = 'METHOD' Identifier method_parameters '{' method_attribute_list '}'
```

```
method_parameters:
    = '(' method_parameter_list ')'
```

```
method_parameter_list:
    = method_parameter
    = method_parameter_list ',' method_parameter
```

```
method_parameter:
    = method_parameter_type Identifier
    = method_parameter_type '&' Identifier
    = method_parameter_type Identifier '[' ']'
```

```
method_parameter_type:
    = 'char'
    = 'unsigned' 'char'
    = 'short'
    = 'unsigned' 'short'
    = 'int'
    = 'unsigned' 'int'
    = 'long'
    = 'unsigned' 'long'
    = 'long' 'long'
    = 'unsigned' 'long' 'long'
    = 'float'
    = 'double'
    = 'time_t'
    = 'DD_STRING'
    = 'DD_ITEM'
```

```
method_attribute_list:
    = method_attribute
    = method_attribute_list method_attribute
```

```
method_attribute:
    = method_definition          /* M */
    = method_access              /* O */
    = method_class               /* O */
    = help                      /* O */
    = label                     /* O */
    = private                   /* O */
    = method_type               /* O */
    = validity                  /* O */
    = visibility                 /* O */
```

```
method_definition:
    = 'DEFINITION' c_compound_statement
```

```
method_access:
    = 'ACCESS' 'OFFLINE' ';'
    = 'ACCESS' 'ONLINE' ';'

```

```

method_class:
    = 'CLASS' method_class_definition ';'

method_class_definition:
    = method_class_keyword
    = method_class_definition '&' method_class_keyword

method_class_keyword:
    = 'ALARM'
    = 'ANALOG_OUTPUT'
    = 'COMPUTATION'
    = 'CONTAINED'
    = 'CORRECTION'
    = 'DEVICE'
    = 'DIAGNOSTIC'
    = 'DISCRETE'
    = 'FREQUENCY'
    = 'HART'
    = 'INPUT'
    = 'LOCAL_DISPLAY'
    = 'OPERATE'
    = 'OUTPUT'
    = 'SERVICE'
    = 'SPECIALIST'
    = 'TUNE'

method_type:
    = 'TYPE' method_return_type

method_return_type:
    = 'char'
    = 'unsigned' 'char'
    = 'short'
    = 'unsigned' 'short'
    = 'int'
    = 'unsigned' 'int'
    = 'long'
    = 'unsigned' 'long'
    = 'long' 'long'
    = 'unsigned' 'long' 'long'
    = 'float'
    = 'double'
    = 'time_t'
    = 'DD_STRING'

```

A.6.25 PROGRAM (void)

A.6.26 RECORD

```

record:
    = 'RECORD' Identifier '{' record_attribute_list '}'

record_attribute_list:
    = record_attribute
    = record_attribute_list record_attribute

record_attribute:
    = label /* M */
    = members /* M */
    = help /* O */
    = private /* O */
    = response_codes /* O */
    = validity /* O */

```

```

= visibility                /* O */
= write_mode                /* O */

```

A.6.27 REFERENCE_ARRAY

Reference_Array has two available keywords, "ITEM_ARRAY" (first option) and "ARRAY" (second option) which may be selected by a profile. The keyword ARRAY shall not be selected if the value_array is used.

If an EDD application supports more than one profile, the keyword ARRAY can be also used instead of ITEM_ARRAY. In this case, the semantical selection is done by attributes.

```

item_array:
  = 'ITEM_ARRAY' 'OF' collection_item_type Identifier
    '{' item_array_attribute_list '}'
  = 'ARRAY' 'OF' collection_item_type Identifier
    '{' item_array_attribute_list '}'

item_array_attribute_list:
  = item_array_attribute
  = item_array_attribute_list item_array_attribute

item_array_attribute:
  = elements                /* M */
  = help                    /* O */
  = label                   /* O */
  = private                 /* O */
  = validity                /* O */
  = visibility              /* O */

elements:
  = 'ELEMENTS' '{' elements_specifier_list '}'

elements_specifier_list:
  = elements_specifier
  = elements_specifier_list elements_specifier

elements_specifier:
  = element
  = 'IF' '(' expr ')' '{' elements_specifier_list '}'
  = 'IF' '(' expr ')' '{' elements_specifier_list '}'
  = 'ELSE' '{' elements_specifier_list '}'
  = 'SELECT' '(' expr ')' '{' elements_selection_list '}'

elements_selection_list:
  = elements_selection
  = elements_selection_list elements_selection

elements_selection:
  = 'CASE' expr ':' elements_specifier_list
  = 'DEFAULT' ':' elements_specifier_list

element:
  = Integer ',' reference_without_call ';'
  = Integer ',' reference_without_call ',' description_string ';'
  = Integer ',' reference_without_call ',' description_string
    ',' help_string ';'

```

A.6.28 Relations

```

refresh_relation:
  = 'REFRESH' Identifier
    '{' refresh_relation_cause ':' refresh_relation_effect '}'

```

```

refresh_relation_cause:
    = refresh_cause_specifier_list

refresh_cause_specifier_list:
    = refresh_cause_specifier
    = refresh_cause_specifier_list refresh_cause_specifier
    = refresh_cause_specifier_list ',' refresh_cause_specifier

refresh_cause_specifier:
    = variable_reference
    = record_reference
    = value_array_reference
    = 'IF' '(' expr ')' '{' refresh_cause_specifier '}'
    = 'IF' '(' expr ')' '{' refresh_cause_specifier '}'
    = 'ELSE' '{' refresh_cause_specifier '}'
    = 'SELECT' '(' expr ')' '{' refresh_cause_selection_list '}'

refresh_cause_selection_list:
    = refresh_cause_selection
    = refresh_cause_selection_list refresh_cause_selection

refresh_cause_selection:
    = 'CASE' expr ':' refresh_cause_specifier
    = 'DEFAULT' ':' refresh_cause_specifier

refresh_relation_effect:
    = refresh_effect_specifier_list

refresh_effect_specifier_list:
    = refresh_effect_specifier
    = refresh_effect_specifier_list refresh_effect_specifier
    = refresh_effect_specifier_list ',' refresh_effect_specifier

refresh_effect_specifier:
    = variable_reference
    = list_reference
    = record_reference
    = value_array_reference
    = 'IF' '(' expr ')' '{' refresh_effect_specifier '}'
    = 'IF' '(' expr ')' '{' refresh_effect_specifier '}'
    = 'ELSE' '{' refresh_effect_specifier '}'
    = 'SELECT' '(' expr ')' '{' refresh_effect_selection_list '}'

refresh_effect_selection_list:
    = refresh_effect_selection
    = refresh_effect_selection_list refresh_effect_selection

refresh_effect_selection:
    = 'CASE' expr ':' refresh_effect_specifier
    = 'DEFAULT' ':' refresh_effect_specifier

unit_relation:
    = 'UNIT' Identifier
    = '{' unit_relation_cause ':' unit_relation_effect '}'

unit_relation_cause:
    = variable_reference_specifier

unit_relation_effect:
    = unit_effect_selection_list

wao_relation:

```



```

    = 'WRITE_AS_ONE' Identifier '{' wao_specifier '}'

wao_specifier:
    = wao_reference_specifier_list

wao_reference_specifier_list:
    = wao_reference_specifier
    = wao_reference_specifier_list wao_reference_specifier
    = wao_reference_specifier_list ',' wao_reference_specifier

wao_reference_specifier:
    = variable_reference
    = record_reference
    = value_array_reference
    = 'IF' '(' expr ')' '{' wao_reference_specifier '}'
    = 'IF' '(' expr ')' '{' wao_reference_specifier '}'
    'ELSE' '{' wao_reference_specifier '}'
    = 'SELECT' '(' expr ')' '{' wao_reference_selection_list '}'

wao_reference_selection_list:
    = wao_reference_selection
    = wao_reference_selection_list wao_reference_selection

wao_reference_selection:
    = 'CASE' expr ':' wao_reference_specifier
    = 'DEFAULT' ':' wao_reference_specifier

unit_effect_selection_list:
    = unit_effect_selection
    = unit_effect_selection_list unit_effect_selection

unit_effect_selection:
    = 'CASE' expr ':' unit_effect_specifier
    = 'DEFAULT' ':' unit_effect_specifier

unit_effect_specifier:
    = variable_reference
    = axis_reference
    = list_reference
    = value_array_reference
    = 'IF' '(' expr ')' '{' unit_effect_specifier '}'
    = 'IF' '(' expr ')' '{' unit_effect_specifier '}'
    'ELSE' '{' unit_effect_specifier '}'
    = 'SELECT' '(' expr ')' '{' unit_effect_selection_list '}'

```

A.6.29 RESPONSE_CODES

```

response_codes_definition:
    = 'RESPONSE_CODES' Identifier '{' response_codes_attribute_list
    '}'

response_codes_attribute_list:
    = response_codes_specifier_list

response_codes_specifier_list:
    = response_codes_specifier
    = response_codes_specifier_list response_codes_specifier

response_codes_specifier:
    = response_code
    = 'IF' '(' expr ')' '{' response_codes_specifier_list '}'
    = 'IF' '(' expr ')' '{' response_codes_specifier_list '}'
    'ELSE' '{' response_codes_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' response_codes_selection_list '}'

```

```

response_codes_selection_list:
    = response_codes_selection
    = response_codes_selection_list response_codes_selection

response_codes_selection:
    = 'CASE' expr ':' response_codes_specifier_list
    = 'DEFAULT' ':' response_codes_specifier_list

response_code:
    = Integer ',' response_code_type ',' description_string ';'
    = Integer ',' response_code_type ',' description_string ','
      help_string ';'

response_code_type:
    = 'SUCCESS'
    = 'MISC_WARNING'
    = 'DATA_ENTRY_WARNING'
    = 'DATA_ENTRY_ERROR'
    = 'MODE_ERROR'
    = 'PROCESS_ERROR'
    = 'MISC_ERROR'

```

A.6.30 SOURCE

```

source:
    = 'SOURCE' Identifier '{' source_attribute_list '}'

source_attribute_list:
    = source_attribute
    = source_attribute_list source_attribute

source_attribute:
    = members /* M */
    = emphasis /* O */
    = source_exit_actions /* O */
    = help /* O */
    = source_init_actions /* O */
    = label /* O */
    = line_color /* O */
    = line_type /* O */
    = source_refresh_actions /* O */
    = validity /* O */
    = visibility /* O */
    = source_y_axis /* O */

source_exit_actions:
    = 'EXIT_ACTIONS' '{' actions_specifier_list '}'

source_init_actions:
    = 'INIT_ACTIONS' '{' actions_specifier_list '}'

source_refresh_actions:
    = 'REFRESH_ACTIONS' '{' actions_specifier_list '}'

source_y_axis:
    = 'Y_AXIS' axis_specifier

```

A.6.31 TEMPLATE

```

template:
    = 'TEMPLATE' Identifier '{' template_attribute_list '}'

template_attribute_list:

```

```

    = template_attribute
    = template_attribute_list template_attribute

template_attribute:
    = template_default_values          /* M */
    = label                            /* M */
    = help                             /* O */
    = validity                         /* O */

template_default_values:
    = 'DEFAULT_VALUES' '{' template_default_value_list '}'

template_default_value_list:
    = template_default_value
    = template_default_value_list template_default_value

template_default_value:
    = reference_without_call '=' expr ';'
    = reference_without_call '=' array_initialization

```

A.6.32 VALUE_ARRAY

```

array:
    = 'ARRAY' Identifier '{' array_attribute_list '}'

array_attribute_list:
    = array_attribute
    = array_attribute_list array_attribute

array_attribute:
    = label                            /* M */
    = array_size                       /* M */
    = array_type                       /* M */
    = help                             /* O */
    = private                         /* O */
    = response_codes                  /* O */
    = validity                        /* O */
    = visibility                      /* O */
    = write_mode                      /* O */

array_size:
    = 'NUMBER_OF_ELEMENTS' Integer ';'

array_type:
    = 'TYPE' Identifier ';'

```

A.6.33 VARIABLE

```

variable:
    = 'VARIABLE' Identifier '{' variable_attribute_list '}'

variable_attribute_list:
    = variable_attribute
    = variable_attribute_list variable_attribute

variable_attribute:
    = variable_class                  /* M */
    = type                           /* M */
    = constant_unit                  /* O */
    = default_value                  /* O */
    = handling                       /* O */
    = height                         /* O */
    = help                           /* O */
    = initial_value                  /* O */
    = label                          /* O */

```

```

= post_edit_actions          /* 0 */
= post_read_actions          /* 0 */
= post_rqstupdate_actions    /* 0 */
= post_userchange_actions    /* 0 */
= post_write_actions         /* 0 */
= pre_edit_actions           /* 0 */
= pre_read_actions           /* 0 */
= pre_write_actions          /* 0 */
= private                    /* 0 */
= refresh_actions            /* 0 */
= response_codes             /* 0 */
= validity                   /* 0 */
= visibility                  /* 0 */
= width                      /* 0 */
= write_mode                  /* 0 */

variable_class:
    = 'CLASS' variable_class_definition ';'

variable_class_definition:
    = variable_class_keyword
    = variable_class_definition '&' variable_class_keyword

variable_class_keyword:
    = 'ALARM'
    = 'ANALOG_INPUT'
    = 'ANALOG_OUTPUT'
    = 'COMPUTATION'
    = 'CONTAINED'
    = 'CORRECTION'
    = 'DEVICE'
    = 'DIAGNOSTIC'
    = 'DIGITAL_INPUT'
    = 'DIGITAL_OUTPUT'
    = 'DISCRETE_INPUT'
    = 'DISCRETE_OUTPUT'
    = 'DYNAMIC'
    = 'FREQUENCY_INPUT'
    = 'FREQUENCY_OUTPUT'
    = 'HART'
    = 'INPUT'
    = 'IS_CONFIG'
    = 'LOCAL'
    = 'LOCAL_DISPLAY'
    = 'OPERATE'
    = 'OPTIONAL'
    = 'OUTPUT'
    = 'SERVICE'
    = 'SPECIALIST'
    = 'TEMPORARY'
    = 'TUNE'

type:
    = 'TYPE' type_specifier

type_specifier:
    = arithmetic_type
    = enumerated_type
    = index_type
    = string_type
    = date_time_type
    = boolean_type
    = object_reference

```

```

arithmetic_type:
    = 'INTEGER' arithmetic_size arithmetic_options
    = 'UNSIGNED_INTEGER' arithmetic_size arithmetic_options
    = 'DOUBLE' arithmetic_options
    = 'FLOAT' arithmetic_options

arithmetic_size:
    =
    = '(' Integer ')'

arithmetic_options:
    = ';'
    = '{' arithmetic_option_list '}'

arithmetic_option_list:
    = arithmetic_option
    = arithmetic_option_list arithmetic_option

arithmetic_option:
    = default_value
    = initial_value
    = display_format
    = edit_format
    = arithmetic_enumerators_specifier
    = maximum_value
    = maximum_value_n
    = minimum_value
    = minimum_value_n
    = scaling_factor

display_format:
    = 'DISPLAY_FORMAT' string_specifier

edit_format:
    = 'EDIT_FORMAT' string_specifier

arithmetic_enumerators_specifier_list:
    = arithmetic_enumerators_specifier
    = arithmetic_enumerators_specifier_list
    arithmetic_enumerators_specifier

arithmetic_enumerators_specifier:
    = arithmetic_enumerator_list
    = 'IF' '(' expr ')' '{' arithmetic_enumerators_specifier_list '}'
    = 'IF' '(' expr ')' '{' arithmetic_enumerators_specifier_list '}'
    = 'ELSE' '{' arithmetic_enumerators_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' arithmetic_enumerators_selection_list '}'

arithmetic_enumerators_selection_list:
    = arithmetic_enumerators_selection
    = arithmetic_enumerators_selection_list
    arithmetic_enumerators_selection

arithmetic_enumerators_selection:
    = 'CASE' expr ':' arithmetic_enumerators_specifier_list
    = 'DEFAULT' ':' arithmetic_enumerators_specifier_list

arithmetic_enumerator_list:
    = arithmetic_enumerator
    = arithmetic_enumerator_list ',' arithmetic_enumerator

arithmetic_enumerator:

```

```

= '{' arithmetic_value ',' description_string '}'
= '{' arithmetic_value ',' description_string ',' help_string '}'

arithmetic_value:
= Integer
= Real

scaling_factor:
= 'SCALING_FACTOR' expr_specifier

maximum_value:
= 'MAX_VALUE' expr_specifier

maximum_value_n:
= 'MAX_VALUE1' expr_specifier
= 'MAX_VALUE2' expr_specifier
= 'MAX_VALUE3' expr_specifier
= 'MAX_VALUE4' expr_specifier
= 'MAX_VALUE5' expr_specifier
= 'MAX_VALUE6' expr_specifier
= 'MAX_VALUE7' expr_specifier
= 'MAX_VALUE8' expr_specifier
= 'MAX_VALUE9' expr_specifier

minimum_value:
= 'MIN_VALUE' expr_specifier

minimum_value_n:
= 'MIN_VALUE1' expr_specifier
= 'MIN_VALUE2' expr_specifier
= 'MIN_VALUE3' expr_specifier
= 'MIN_VALUE4' expr_specifier
= 'MIN_VALUE5' expr_specifier
= 'MIN_VALUE6' expr_specifier
= 'MIN_VALUE7' expr_specifier
= 'MIN_VALUE8' expr_specifier
= 'MIN_VALUE9' expr_specifier

enumerated_type:
= 'ENUMERATED' enumerated_size '{' enumerated_option_list
  '}'
= 'BIT_ENUMERATED' enumerated_size '{'
  bit_enumerated_option_list '}'

enumerated_size:
=
= '(' Integer ')'

enumerated_option_list:
= enumerated_option
= enumerated_option_list enumerated_option

enumerated_option:
= default_value
= initial_value
= enumerators_specifier

enumerators_specifier_list:
= enumerators_specifier
= enumerators_specifier_list enumerators_specifier

enumerators_specifier:
= integral_enumerator_list

```

```

= 'IF' '(' expr ')' '{' enumerators_specifier_list '}'
= 'IF' '(' expr ')' '{' enumerators_specifier_list '}'
  'ELSE' '{' enumerators_specifier_list '}'
= 'SELECT' '(' expr ')' '{' enumerators_selection_list '}'

enumerators_selection_list:
= enumerators_selection
= enumerators_selection_list enumerators_selection

enumerators_selection:
= 'CASE' expr ':' enumerators_specifier_list
= 'DEFAULT' ':' enumerators_specifier_list

integral_enumerator_list:
= integral_enumerator
= integral_enumerator_list ',' integral_enumerator

integral_enumerator:
= '{' Integer ',' description_string '}'
= '{' Integer ',' description_string ',' help_string '}'

description_string:
= string

help_string:
= string

bit_enumerated_option_list:
= bit_enumerated_option
= bit_enumerated_option_list bit_enumerated_option

bit_enumerated_option:
= default_value
= initial_value
= bit_enumerators_specifier

bit_enumerators_specifier_list:
= bit_enumerators_specifier
= bit_enumerators_specifier_list bit_enumerators_specifier

bit_enumerators_specifier:
= bit_enumerator_list
= 'IF' '(' expr ')' '{' bit_enumerators_specifier_list '}'
= 'IF' '(' expr ')' '{' bit_enumerators_specifier_list '}'
  'ELSE' '{' bit_enumerators_specifier_list '}'
= 'SELECT' '(' expr ')' '{' bit_enumerators_selection_list '}'

bit_enumerators_selection_list:
= bit_enumerators_selection
= bit_enumerators_selection_list bit_enumerators_selection

bit_enumerators_selection:
= 'CASE' expr ':' bit_enumerators_specifier_list
= 'DEFAULT' ':' bit_enumerators_specifier_list

bit_enumerator_list:
= bit_enumerator
= bit_enumerator_list ',' bit_enumerator

bit_enumerator:
= '{' Integer ',' string '}'
= '{' Integer ',' string ',' help_string '}'
= '{' Integer ',' string ',' variable_class_definition '}'

```

```

= '{' Integer ',' string ',' status_class '}'
= '{' Integer ',' string ',' string ',' variable_class_definition '}'
= '{' Integer ',' string ',' string ',' status_class '}'
= '{' Integer ',' string ',' string ',' method_reference '}'
= '{' Integer ',' string ',' variable_class_definition ','
  status_class '}'
= '{' Integer ',' string ',' variable_class_definition ','
  method_reference '}'
= '{' Integer ',' string ',' status_class ',' method_reference '}'
= '{' Integer ',' string ',' string ',' variable_class_definition ','
  status_class '}'
= '{' Integer ',' string ',' string ',' variable_class_definition ','
  method_reference '}'
= '{' Integer ',' string ',' string ',' status_class ','
  method_reference '}'
= '{' Integer ',' string ',' variable_class_definition ',' status_class
  ',' method_reference '}'
= '{' Integer ',' string ',' string ',' variable_class_definition ','
  status_class ',' method_reference '}'

status_class:
  = status_class_definition

status_class_definition:
  = status_class_keyword
  = status_class_definition '&' status_class_keyword

status_class_keyword:
  = 'HARDWARE'
  = 'SOFTWARE'
  = 'PROCESS'
  = 'MODE'
  = 'DATA'
  = 'MISC'
  = 'EVENT'
  = 'STATE'
  = 'SELF_CORRECTING'
  = 'CORRECTABLE'
  = 'UNCORRECTABLE'
  = 'SUMMARY'
  = 'DETAIL'
  = 'MORE'
  = 'COMM_ERROR'
  = 'IGNORE_IN_TEMPORARY_MASTER'
  = 'BAD'
  = 'INFO'
  = 'WARNING'
  = 'IGNORE_IN_HOST'
  = 'ERROR'
  = 'IGNORE_IN_HANDHELD'
  = 'DV' '(' output-mode ')'
  = 'TV' '(' output-mode ')'
  = 'AO' '(' output-mode ')'
  = 'ALL' '(' output-mode ')'
  = 'DVn' '(' output-mode ')'
  = 'TVn' '(' output-mode ')'
  = 'AOn' '(' output-mode ')'

output-mode:
  = reliability '&' mode
  = mode '&' reliability

reliability:

```



```
= 'GOOD'
= 'MARGINAL'
= 'BAD'

mode:
= 'AUTO'
= 'MANUAL'

DVn:
= 'DV1'
= 'DV2'
= 'DV3'
= 'DV4'

TVn:
= 'TV1'
= 'TV2'
= 'TV3'
= 'TV4'
= 'TV5'
= 'TV6'
= 'TV7'
= 'TV8'
= 'TV9'
= 'TV10'
= 'TV11'
= 'TV12'
= 'TV13'
= 'TV14'
= 'TV15'
= 'TV16'
= 'TV17'
= 'TV18'
= 'TV19'
= 'TV20'

AOn:
= 'AO1'
= 'AO2'
= 'AO3'
= 'AO4'
= 'AO5'

index_type:
= 'INDEX' item_array_reference ';'
= 'INDEX' '(' Integer ')' item_array_reference ';'

string_type:
= 'ASCII' string_size string_options
= 'BITSTRING' string_size string_options
= 'EUC' string_size string_options
= 'PACKED_ASCII' string_size string_options
= 'PASSWORD' string_size string_options
= 'VISIBLE' string_size string_options
= 'OCTET' string_size string_options

string_size:
= '(' Integer ') '

string_options:
= ';'
= '{' string_option_list '}'
```

```

string_option_list:
    = string_option
    = string_option_list string_option

string_option:
    = string_default_value
    = string_initial_value
    = string_enumerators_specifier

string_enumerators_specifier_list:
    = string_enumerators_specifier
    = string_enumerators_specifier_list string_enumerators_specifier

string_enumerators_specifier:
    = string_enumerator_list
    = 'IF' '(' expr ')' '{' string_enumerators_specifier_list '}'
    = 'IF' '(' expr ')' '{' string_enumerators_specifier_list '}'
    'ELSE' '{' string_enumerators_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' string_enumerators_selection_list '}'

string_enumerators_selection_list:
    = string_enumerators_selection
    = string_enumerators_selection_list string_enumerators_selection

string_enumerators_selection:
    = 'CASE' expr ':' string_enumerators_specifier_list
    = 'DEFAULT' ':' string_enumerators_specifier_list

string_enumerator_list:
    = string_enumerator
    = string_enumerator_list ',' string_enumerator

string_enumerator:
    = '{' string ',' description_string '}'
    = '{' string ',' description_string ',' help_string '}'

string_default_value:
    = 'DEFAULT_VALUE' string_specifier

string_initial_value:
    = 'INITIAL_VALUE' string ';'

date_time_type:
    = 'DATE' date_time_options
    = 'DATE AND TIME' date_time_options
    = 'DURATION' date_time_options
    = 'TIME' date_time_options
    = 'TIME_VALUE' date_time_size time_value_options

date_time_size:
    =
    = '(' Integer ')'

date_time_options:
    = ';'
    = '{' date_time_option_list '}'

date_time_option_list:
    = date_time_option
    = date_time_option_list date_time_option

date_time_option:
    = default_value

```

```
= initial_value
= maximum_value
= maximum_value_n
= minimum_value
= minimum_value_n

time_value_options:
= ';'
= '{' time_value_option_list '}'

time_value_option_list:
= time_value_option
= time_value_option_list time_value_option

time_value_option:
= default_value
= display_format
= edit_format
= initial_value
= maximum_value
= maximum_value_n
= minimum_value
= minimum_value_n
= time_format
= time_scale

time_format:
= 'TIME_FORMAT' string_specifier

time_scale:
= 'TIME_SCALE' time_scale_specifier

time_scale_specifier:
= time_scale_keyword ';'
= 'IF' '(' expr ')' '{' time_scale_specifier '}'
= 'IF' '(' expr ')' '{' time_scale_specifier '}'
  'ELSE' '{' time_scale_specifier '}'
= 'SELECT' '(' expr ')' '{' time_scale_selection_list '}'

time_scale_selection_list:
= time_scale_selection
= time_scale_selection_list time_scale_selection

time_scale_selection:
= 'CASE' expr ':' time_scale_specifier
= 'DEFAULT' ':' time_scale_specifier

time_scale_keyword:
= 'SECONDS'
= 'MINUTES'
= 'HOURS'

object_reference:
= 'OBJECT_REFERENCE' ';'

boolean_type:
= 'BOOLEAN' ';'

constant_unit:
= 'CONSTANT_UNIT' string_specifier

default_value:
= 'DEFAULT_VALUE' expr_specifier
```

```

initial_value:
    = 'INITIAL_VALUE' expr ';'

pre_edit_actions:
    = 'PRE_EDIT_ACTIONS' '{' actions_specifier_list '}'

post_edit_actions:
    = 'POST_EDIT_ACTIONS' '{' actions_specifier_list '}'

pre_read_actions:
    = 'PRE_READ_ACTIONS' '{' actions_specifier_list '}'

post_read_actions:
    = 'POST_READ_ACTIONS' '{' actions_specifier_list '}'

pre_write_actions:
    = 'PRE_WRITE_ACTIONS' '{' actions_specifier_list '}'

post_write_actions:
    = 'POST_WRITE_ACTIONS' '{' actions_specifier_list '}'

refresh_actions:
    = 'REFRESH_ACTIONS' '{' actions_specifier_list '}'

actions_specifier_list:
    = actions_specifier
    = actions_specifier_list ',' actions_specifier

actions_specifier:
    = method_specification
    = 'IF' '(' expr ')' '{' actions_specifier_list '}'
    = 'IF' '(' expr ')' '{' actions_specifier_list '}'
    = 'ELSE' '{' actions_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' actions_selection_list '}'

actions_selection_list:
    = actions_selection
    = actions_selection_list actions_selection

actions_selection:
    = 'CASE' expr ':' actions_specifier_list
    = 'DEFAULT' ':' actions_specifier_list

method_specification:
    = method_reference
    = method_definition

expr_specifier:
    = expr ';'
    = 'IF' '(' expr ')' '{' expr_specifier '}'
    = 'IF' '(' expr ')' '{' expr_specifier '}'
    = 'ELSE' '{' expr_specifier '}'
    = 'SELECT' '(' expr ')' '{' expr_selection_list '}'

expr_selection_list:
    = expr_selection
    = expr_selection_list expr_selection

expr_selection:
    = 'CASE' expr ':' expr_specifier
    = 'DEFAULT' ':' expr_specifier

```

```
post_rqstupdate_actions:
    = 'POST_RQSTUPDATE_ACTIONS' '{' actions_specifier_list '}'
```

```
post_userchange_actions:
    = 'POST_USERCHANGE_ACTIONS' '{' actions_specifier_list '}'
```

A.6.34 VARIABLE_LIST

```
variable_list:
    = 'VARIABLE_LIST' Identifier '{' variable_list_attribute_list '}'
```

```
variable_list_attribute_list:
    = variable_list_attribute
    = variable_list_attribute_list variable_list_attribute
```

```
variable_list_attribute:
    = members                /* M */
    = help                    /* O */
    = label                   /* O */
    = response_codes          /* O */
```

A.6.35 WAVEFORM

```
waveform:
    = 'WAVEFORM' Identifier '{' waveform_attribute_list '}'
```

```
waveform_attribute_list:
    = waveform_attribute
    = waveform_attribute_list waveform_attribute
```

```
waveform_attribute:
    = waveform_type           /* M */
    = emphasis                 /* O */
    = waveform_exit_actions   /* O */
    = handling                 /* O */
    = help                     /* O */
    = waveform_init_actions   /* O */
    = waveform_key_points     /* O */
    = label                    /* O */
    = line_color               /* O */
    = line_type                /* O */
    = waveform_refresh_actions /* O */
    = validity                 /* O */
    = visibility               /* O */
    = waveform_y_axis          /* O */
```

```
waveform_type:
    = 'TYPE' waveform_type_specifier
```

```
waveform_type_specifier:
    = waveform_yt_type
    = waveform_xy_type
    = waveform_horizontal_type
    = waveform_vertical_type
```

```
waveform_yt_type:
    = 'YT' '{' waveform_yt_type_option_list '}'
```

```
waveform_yt_type_option_list:
    = waveform_yt_type_option
    = waveform_yt_type_option_list waveform_yt_type_option
```

```
waveform_yt_type_option:
    = yt_x_initial            /* M */
    = yt_x_increment          /* M */
```

```

= waveform_y_values          /* M */
= number_of_points           /* O */

number_of_points:
    = 'NUMBER_OF_POINTS' expr_specifier

yt_x_initial:
    = 'X_INITIAL' expr_specifier

yt_x_increment:
    = 'X_INCREMENT' expr_specifier

waveform_y_values:
    = 'Y_VALUES' '{' values_option_list '}'

waveform_x_values:
    = 'X_VALUES' '{' values_option_list '}'

values_option_list:
    = values_option
    = values_option_list ',' values_option

values_option:
    = reference_without_call
    = Integer
    = Real

waveform_xy_type:
    = 'XY' '{' waveform_xy_type_option_list '}'

waveform_xy_type_option_list:
    = waveform_xy_type_option
    = waveform_xy_type_option_list waveform_xy_type_option

waveform_xy_type_option:
    = waveform_x_values          /* M */
    = waveform_y_values          /* M */
    = number_of_points           /* O */

waveform_horizontal_type:
    = 'HORIZONTAL' '{' waveform_y_values '}'

waveform_vertical_type:
    = 'VERTICAL' '{' waveform_x_values '}'

waveform_y_axis:
    = 'Y_AXIS' axis_specifier

axis_specifier:
    = axis_reference ';'
    = 'IF' '(' expr ')' '{' axis_specifier '}'
    = 'IF' '(' expr ')' '{' axis_specifier '}'
    = 'ELSE' '{' axis_specifier '}'
    = 'SELECT' '(' expr ')' '{' axis_selection_list '}'

axis_selection_list:
    = axis_selection
    = axis_selection_list axis_selection

axis_selection:
    = 'CASE' expr ':' axis_specifier
    = 'DEFAULT' ':' axis_specifier

```

```

waveform_exit_actions:
    = 'EXIT_ACTIONS' '{' actions_specifier_list '}'

waveform_init_actions:
    = 'INIT_ACTIONS' '{' actions_specifier_list '}'

waveform_key_points:
    = 'KEY_POINTS' '{' key_points_options_list '}'

key_points_options_list:
    = key_points_options
    = key_points_options_list key_points_options

key_points_options:
    = waveform_x_values                /* M */
    = waveform_y_values                /* M */

waveform_refresh_actions:
    = 'REFRESH_ACTIONS' '{' actions_specifier_list '}'

```

A.6.36 Common attributes

```

emphasis:
    = 'EMPHASIS' boolean_specifier

boolean_specifier:
    = boolean ';'
    = 'IF' '(' expr ')' '{' boolean_specifier '}'
    = 'IF' '(' expr ')' '{' boolean_specifier '}'
    = 'ELSE' '{' boolean_specifier '}'
    = 'SELECT' '(' expr ')' '{' boolean_selection_list '}'

boolean_selection_list:
    = boolean_selection
    = boolean_selection_list boolean_selection

boolean_selection:
    = 'CASE' expr ':' boolean_specifier
    = 'DEFAULT' ':' boolean_specifier

boolean:
    = 'FALSE'
    = 'TRUE'

handling:
    = 'HANDLING' handling_specifier

handling_specifier:
    = handling_definition ';'
    = 'IF' '(' expr ')' '{' handling_specifier '}'
    = 'IF' '(' expr ')' '{' handling_specifier '}'
    = 'ELSE' '{' handling_specifier '}'
    = 'SELECT' '(' expr ')' '{' handling_selection_list '}'

handling_selection_list:
    = handling_selection
    = handling_selection_list handling_selection

handling_selection:
    = 'CASE' expr ':' handling_specifier
    = 'DEFAULT' ':' handling_specifier

handling_definition:
    = handling_keyword

```

```

    = handling_definition '&' handling_keyword

handling_keyword:
    = 'READ'
    = 'WRITE'

height:
    = 'HEIGHT' display_size_keyword ';'

display_size_keyword:
    = 'XXX_SMALL'
    = 'XX_SMALL'
    = 'X_SMALL'
    = 'SMALL'
    = 'MEDIUM'
    = 'XX_LARGE'
    = 'X_LARGE'
    = 'LARGE'

help:
    = 'HELP' string_specifier

string_specifier:
    = string ';'
    = octet_string ';'
    = 'IF' '(' expr ')' '{' string_specifier '}'
    = 'IF' '(' expr ')' '{' string_specifier '}'
    = 'ELSE' '{' string_specifier '}'
    = 'SELECT' '(' expr ')' '{' string_selection_list '}'

string_selection_list:
    = string_selection
    = string_selection_list string_selection

string_selection:
    = 'CASE' expr ':' string_specifier
    = 'DEFAULT' ':' string_specifier

string:
    = string_list

string_list:
    = string_terminal
    = string_list '+' string_terminal

string_terminal:
    = string_constant
    = variable_reference

string_literal:
    = String
    = string_literal String

string_constant:
    = string_literal
    = string_attribute_reference
    = enumerated_variable_reference '(' Integer ')'
    = dictionary_reference

octet_string:
    = '{' octet_string_list '}'

octet_string_list:

```



```

    = octet_string_element
    = octet_string_list ',' octet_string_element

octet_string_element:
    = Integer
    = Character

label:
    = 'LABEL' string_specifier

identity:
    = 'IDENTITY' filename ';'

line_color:
    = 'LINE_COLOR' line_color_specifier

line_color_specifier:
    = color ';'
    = 'IF' '(' expr ')' '{' line_color_specifier '}'
    = 'IF' '(' expr ')' '{' line_color_specifier '}'
    'ELSE' '{' line_color_specifier '}'
    = 'SELECT' '(' expr ')' '{' line_color_selection_list '}'

line_color_selection_list:
    = line_color_selection
    = line_color_selection_list line_color_selection

line_color_selection:
    = 'CASE' expr ':' line_color_specifier
    = 'DEFAULT' ':' line_color_specifier

color:
    = Integer
    = variable_reference

line_type:
    = 'LINE_TYPE' line_type_specifier

line_type_specifier:
    = line_type_keyword ';'
    = 'IF' '(' expr ')' '{' line_type_specifier '}'
    = 'IF' '(' expr ')' '{' line_type_specifier '}'
    'ELSE' '{' line_type_specifier '}'
    = 'SELECT' '(' expr ')' '{' line_type_selection_list '}'

line_type_selection_list:
    = line_type_selection
    = line_type_selection_list line_type_selection

line_type_selection:
    = 'CASE' expr ':' line_type_specifier
    = 'DEFAULT' ':' line_type_specifier

line_type_keyword:
    = 'DATA'
    = 'DATA1'
    = 'DATA2'
    = 'DATA3'
    = 'DATA4'
    = 'DATA5'
    = 'DATA6'
    = 'DATA7'
    = 'DATA8'

```

```

= 'DATA9'
= 'LOW_LOW_LIMIT'
= 'LOW_LIMIT'
= 'HIGH_LIMIT'
= 'HIGH_HIGH_LIMIT'
= 'TRANSPARENT'

members:
= 'MEMBERS' '{' members_specifier_list '}'

members_specifier_list:
= members_specifier
= members_specifier_list members_specifier

members_specifier:
= member
= 'IF' '(' expr ')' '{' members_specifier_list '}'
= 'IF' '(' expr ')' '{' members_specifier_list '}'
'ELSE' '{' members_specifier_list '}'
= 'SELECT' '(' expr ')' '{' members_selection_list '}'

members_selection_list:
= members_selection
= members_selection_list members_selection

members_selection:
= 'CASE' expr ':' members_specifier_list
= 'DEFAULT' ':' members_specifier_list

member:
= Identifier ',' reference_without_call ';'
= Identifier ',' reference_without_call ',' string ';'
= Identifier ',' reference_without_call ',' string ',' string ';'

private:
= 'PRIVATE' boolean_specifier

response_codes:
= 'RESPONSE_CODES' '{' response_codes_specifier_list '}'
= 'RESPONSE_CODES' response_code_reference_specifier

response_code_reference_specifier:
= response_code_reference ';'
= 'IF' '(' expr ')' '{' response_code_reference_specifier '}'
= 'IF' '(' expr ')' '{' response_code_reference_specifier
'}'
'ELSE' '{' response_code_reference_specifier '}'
= 'SELECT' '(' expr ')' '{'
response_code_reference_selection_list '}'

response_code_reference_selection_list:
= response_code_reference_selection
= response_code_reference_selection_list
response_code_reference_selection

response_code_reference_selection:
= 'CASE' expr ':' response_code_reference_specifier
= 'DEFAULT' ':' response_code_reference_specifier

validity:
= 'VALIDITY' boolean_specifier

visibility:

```

```

    = 'VISIBILITY' boolean_specifier

width:
    = 'WIDTH' display_size_keyword ';'

write_mode:
    = 'WRITE_MODE' write_mode_keyword ';'

write_mode_keyword:
    = 'MODE_OUT_OF_SERVICE'
    = 'MODE_INITIALIZATION'
    = 'MODE_LOCAL_OVERRIDE'
    = 'MODE_MANUAL'
    = 'MODE_AUTOMATIC'
    = 'MODE_CASCADE'
    = 'MODE_REMOTE_CASCADE'
    = 'MODE_REMOTE_OUTPUT'

```

A.6.37 Expression

```

primary_expr:
    = Integer
    = Real
    = String
    = octet_string
    = variable_reference
    = 'ARRAY_INDEX'
    = 'LINEAR'
    = 'LOGARITHMIC'
    = 'PI'
    = 'TRUE'
    = 'FALSE'
    = 'NULL'
    = 'FIRST'
    = 'LAST'
    = 'CHILD'
    = 'PARENT'
    = 'NEXT'
    = 'PREV'
    = 'SELF'
    = '(' expr ')'

postfix_expr:
    = primary_expr
    = Identifier '(' ')'
    = Identifier '(' c_argument_expr_list ')'
    = axis_reference '.' axis_attribute_selector
    = blob_reference '.' blob_attribute_selector
    = list_reference '.' list_attribute_selector
    = variable_reference '.' variable_attribute_selector
    = array_reference '.' value_array_attribute_selector

axis_attribute_selector:
    = 'MAX_VALUE'
    = 'MIN_VALUE'
    = 'SCALING'
    = 'VIEW_MAX'
    = 'VIEW_MIN'

blob_attribute_selector:
    = 'COUNT'
    = 'IDENTITY'

list_attribute_selector:

```

```
= 'CAPACITY'
= 'COUNT'
= 'FIRST'
= 'LAST'
```

```
value_array_attribute_selector:
    = 'NUMBER_OF_ELEMENTS'
```

```
variable_attribute_selector:
```

```
    = 'DEFAULT_VALUE'
    = 'INITIAL_VALUE'
    = 'MAX_VALUE'
    = 'MAX_VALUE1'
    = 'MAX_VALUE2'
    = 'MAX_VALUE3'
    = 'MAX_VALUE4'
    = 'MAX_VALUE5'
    = 'MAX_VALUE6'
    = 'MAX_VALUE7'
    = 'MAX_VALUE8'
    = 'MAX_VALUE9'
    = 'MIN_VALUE'
    = 'MIN_VALUE1'
    = 'MIN_VALUE2'
    = 'MIN_VALUE3'
    = 'MIN_VALUE4'
    = 'MIN_VALUE5'
    = 'MIN_VALUE6'
    = 'MIN_VALUE7'
    = 'MIN_VALUE8'
    = 'MIN_VALUE9'
    = 'SCALING_FACTOR'
    = 'VARIABLE_STATUS'
```

```
unary_expr:
    = postfix_expr
    = unary_operator unary_expr
```

```
unary_operator:
```

```
= '+'
= '-'
= '~'
= '!'
```

```
multiplicative_expr:
```

```
    = unary_expr
    = multiplicative_expr '*' unary_expr
    = multiplicative_expr '/' unary_expr
    = multiplicative_expr '%' unary_expr
```

```
additive_expr:
```

```
    = multiplicative_expr
    = additive_expr '+' multiplicative_expr
    = additive_expr '-' multiplicative_expr
```

```
shift_expr:
```

```
    = additive_expr
    = shift_expr '<<' additive_expr
    = shift_expr '>>' additive_expr
```

```
relational_expr:
```

```
    = shift_expr
    = relational_expr '<' shift_expr
```

```
= relational_expr '>' shift_expr  
= relational_expr '<=' shift_expr  
= relational_expr '>=' shift_expr
```

```
equality_expr:  
= relational_expr  
= equality_expr '==' relational_expr  
= equality_expr '!=' relational_expr
```

```
and_expr:  
= equality_expr  
= and_expr '&' equality_expr
```

```
exclusive_or_expr:  
= and_expr  
= exclusive_or_expr '^' and_expr
```

```
inclusive_or_expr:  
= exclusive_or_expr  
= inclusive_or_expr '|' exclusive_or_expr
```

```
logical_and_expr:  
= inclusive_or_expr  
= logical_and_expr '&&' inclusive_or_expr
```

```
logical_or_expr:  
= logical_and_expr  
= logical_or_expr '||' logical_and_expr
```

```
conditional_expr:  
= logical_or_expr  
= logical_or_expr '?' expr ':' conditional_expr
```

```
expr:  
= conditional_expr
```

A.6.38 C-Grammar

```
c_primary_expr:  
= c_char  
= c_dictionary  
= c_identifier  
= c_integer  
= c_real  
= c_string  
= c_reference  
= '(' c_expr ')'
```

```
c_char:  
= Character
```

```
c_dictionary:  
= '[' Identifier ']'
```

```
c_identifier:  
= Identifier
```

```
c_integer:  
= Integer
```

```
c_real:  
= Real
```

```
c_string:
```

```

    = c_string_literal

c_string_literal:
    = string_literal

c_reference:
    = reference

c_postfix_expr:
    = c_primary_expr
    = c_postfix_expr '[' c_expr ']'
    = Identifier '(' ')'
    = Identifier '(' c_argument_expr_list ')'
    = c_postfix_expr '.' Identifier
    = c_postfix_expr '.' selector
    = c_postfix_expr '++'
    = c_postfix_expr '--'

c_argument_expr_list:
    = c_assignment_expr
    = c_argument_expr_list ',' c_assignment_expr

c_unary_expr:
    = c_postfix_expr
    = '++' c_unary_expr
    = '--' c_unary_expr
    = c_unary_operator c_postfix_expr

c_unary_operator:
    = '+'
    = '-'
    = '~'
    = '!'

c_multiplicative_expr:
    = c_unary_expr
    = c_multiplicative_expr '*' c_unary_expr
    = c_multiplicative_expr '/' c_unary_expr
    = c_multiplicative_expr '%' c_unary_expr

c_additive_expr:
    = c_multiplicative_expr
    = c_additive_expr '+' c_multiplicative_expr
    = c_additive_expr '-' c_multiplicative_expr

c_shift_expr:
    = c_additive_expr
    = c_shift_expr '<<' c_additive_expr
    = c_shift_expr '>>' c_additive_expr

c_relational_expr:
    = c_shift_expr
    = c_relational_expr '<' c_shift_expr
    = c_relational_expr '>' c_shift_expr
    = c_relational_expr '<=' c_shift_expr
    = c_relational_expr '>=' c_shift_expr

c_equality_expr:
    = c_relational_expr
    = c_equality_expr '==' c_relational_expr
    = c_equality_expr '!=' c_relational_expr

c_and_expr:

```

```
= c_equality_expr
= c_and_expr '&' c_equality_expr

c_exclusive_or_expr:
= c_and_expr
= c_exclusive_or_expr '^' c_and_expr

c_inclusive_or_expr:
= c_exclusive_or_expr
= c_inclusive_or_expr '|' c_exclusive_or_expr

c_logical_and_expr:
= c_inclusive_or_expr
= c_logical_and_expr '&&' c_inclusive_or_expr

c_logical_or_expr:
= c_logical_and_expr
= c_logical_or_expr '||' c_logical_and_expr

c_conditional_expr:
= c_logical_or_expr
= c_logical_or_expr '?' c_logical_or_expr ':' c_conditional_expr

c_assignment_expr:
= c_conditional_expr
= c_unary_expr c_assignment_operator c_assignment_expr

c_assignment_operator:
= '='
= '*='
= '/='
= '%='
= '+='
= '-='
= '>>='
= '<<='
= '&='
= '^='
= '|='

c_expr:
= c_assignment_expr
= c_expr ',' c_assignment_expr

c_constant_expr:
= c_conditional_expr

c_declaration:
= c_type_specifier c_declarator_list ';'

c_type_specifier:
= 'char'
= 'signed' 'char'
= 'unsigned' 'char'
= 'short'
= 'signed' 'short'
= 'unsigned' 'short'
= 'int'
= 'signed' 'int'
= 'unsigned' 'int'
= 'long'
= 'signed' 'long'
= 'unsigned' 'long'
```

```

= 'long' 'long'
= 'signed' 'long' 'long'
= 'unsigned' 'long' 'long'
= 'float'
= 'double'
= 'time_t'
= 'DD_STRING'

c_declarator_list:
= c_declarator
= c_declarator_list ',' c_declarator

c_declarator:
= Identifier
= Identifier c_declarator_array_specifier_list

c_declarator_array_specifier_list:
= c_declarator_array_specifier
= c_declarator_array_specifier_list c_declarator_array_specifier

c_declarator_array_specifier:
= '[' ']'
= '[' c_constant_expr ']'

c_statement:
= c_labeled_statement
= c_compound_statement
= c_expr_statement
= c_selection_statement
= c_iteration_statement
= c_jump_statement

c_labeled_statement:
= 'case' c_constant_expr ':' c_statement
= 'default' ':' c_statement

c_compound_statement:
= '{ ' '}'
= '{' c_statement_list '}'
= '{' c_declaration_list '}'
= '{' c_declaration_list c_statement_list '}'

c_declaration_list:
= c_declaration
= c_declaration_list c_declaration

c_statement_list:
= c_statement
= c_statement_list c_statement

c_expr_statement:
= ';'
= c_expr ';'

c_selection_statement:
= 'if' '(' c_expr ')' c_statement
= 'if' '(' c_expr ')' c_statement 'else' c_statement
= 'switch' '(' c_expr ')' c_statement

c_iteration_statement:
= 'do' c_statement 'while' '(' c_expr ')' ';'
= 'while' '(' c_expr ')' c_statement
= 'for' '(' ';' ';' ')'

```



```

c_statement
= 'for' '('           ';'           ';' c_expr ')'
c_statement
= 'for' '('           ';' c_expr ';'           ')'
c_statement
= 'for' '('           ';' c_expr ';' c_expr ')'
c_statement
= 'for' '(' c_expr ';'           ';'           ')'
c_statement
= 'for' '(' c_expr ';'           ';' c_expr ')'
c_statement
= 'for' '(' c_expr ';' c_expr ';'           ')'
c_statement
= 'for' '(' c_expr ';' c_expr ';' c_expr ')'
c_statement

```

```

c_jump_statement:
= 'continue' ';'
= 'break' ';'
= 'return' ';'
= 'return' c_expr ';'

```

A.6.39 Redefinition

```

array_redefinition:
= 'DELETE' 'ARRAY' Identifier ';'
= 'REDEFINE' 'ARRAY' Identifier '{' array_attribute_list '}'
= 'ARRAY' Identifier '{' array_attribute_redefinition_list '}'

array_attribute_redefinition_list:
= array_attribute_redefinition
= array_attribute_redefinition_list array_attribute_redefinition

array_attribute_redefinition:
= array_size_redefinition
= array_type_redefinition
= help_redefinition
= private_redefinition
= required_label_redefinition
= response_codes_reference_redefinition
= validity_redefinition
= visibility_redefinition
= write_mode_redefinition

array_size_redefinition:
= 'REDEFINE' array_size

array_type_redefinition:
= 'REDEFINE' array_type

axis_redefinition:
= 'DELETE' 'AXIS' Identifier ';'
= 'REDEFINE' 'AXIS' Identifier '{' '}'
= 'REDEFINE' 'AXIS' Identifier '{' axis_attribute_list '}'
= 'AXIS' Identifier '{' axis_attribute_redefinition_list '}'

axis_attribute_redefinition_list:
= axis_attribute_redefinition
= axis_attribute_redefinition_list axis_attribute_redefinition

axis_attribute_redefinition:
= constant_unit_redefinition
= help_redefinition

```

```

= optional_label_redefinition
= maximum_value_redefinition
= minimum_value_redefinition
= axis_scaling_redefinition

axis_scaling_redefinition:
= 'DELETE' 'SCALING' ';'
= 'REDEFINE' axis_scaling

blob_redefinition:
= 'DELETE' 'BLOB' Identifier ';'
= 'REDEFINE' 'BLOB' Identifier '{' '}'
= 'REDEFINE' 'BLOB' Identifier '{' blob_attribute_list '}'
= 'BLOB' Identifier '{' blob_attribute_redefinition_list '}'

blob_attribute_redefinition_list:
= blob_attribute_redefinition
= blob_attribute_redefinition_list blob_attribute_redefinition

blob_attribute_redefinition:
= required_handling_redefinition
= help_redefinition
= identity_redefinition
= optional_label_redefinition

identity_redefinition:
= 'DELETE' 'IDENTITY' ';'
= 'REDEFINE' identity

block_redefinition:
= 'DELETE' 'BLOCK' Identifier ';'
= 'REDEFINE' 'BLOCK' Identifier '{' '}'
= 'REDEFINE' 'BLOCK' Identifier '{' block_attribute_list '}'
= 'BLOCK' Identifier '{' block_attribute_redefinition_list '}'

block_attribute_redefinition_list:
= block_attribute_redefinition
= block_attribute_redefinition_list block_attribute_redefinition

block_attribute_redefinition:
= block_a_characteristics_redefinition
= required_label_redefinition
= block_a_parameters_redefinition
= block_a_axis_items_redefinition
= block_a_chart_items_redefinition
= block_a_collection_items_redefinition
= block_a_edit_display_items_redefinition
= block_a_file_items_redefinition
= block_a_graph_items_redefinition
= block_a_grid_items_redefinition
= help_redefinition
= block_a_image_items_redefinition
= block_a_item_array_items_redefinition
= block_a_list_items_redefinition
= block_a_local_parameters_redefinition
= block_a_menu_items_redefinition
= block_a_method_items_redefinition
= block_a_parameter_lists_redefinition
= block_a_plugin_items_redefinition
= block_a_refresh_items_redefinition
= block_a_source_items_redefinition
= block_a_unit_items_redefinition
= block_a_wao_items_redefinition

```

```
= block_a_waveform_items_redefinition
= block_a_charts_redefinition
= block_a_files_redefinition
= block_a_lists_redefinition
= block_a_graphs_redefinition
= block_a_grids_redefinition
= block_a_menus_redefinition
= block_a_methods_redefinition
= block_a_plugins_redefinition
= block_b_number_redefinition
= block_b_type_redefinition

block_a_characteristics_redefinition:
    = 'REDEFINE' block_a_characteristics

block_a_parameters_redefinition:
    = 'PARAMETERS' '{' parameter_redefinition_list '}'

parameter_redefinition_list:
    = parameter_redefinition
    = parameter_redefinition_list parameter_redefinition

parameter_redefinition:
    = 'REDEFINE' member
    = 'ADD' member

block_a_axis_items_redefinition:
    = 'DELETE' 'AXIS_ITEMS' ';'
    = 'REDEFINE' block_a_axis_items

block_a_chart_items_redefinition:
    = 'DELETE' 'CHART_ITEMS' ';'
    = 'REDEFINE' block_a_chart_items

block_a_collection_items_redefinition:
    = 'DELETE' 'COLLECTION_ITEMS' ';'
    = 'REDEFINE' block_a_collection_items

block_a_edit_display_items_redefinition:
    = 'DELETE' 'EDIT_DISPLAY_ITEMS' ';'
    = 'REDEFINE' block_a_edit_display_items

block_a_file_items_redefinition:
    = 'DELETE' 'FILE_ITEMS' ';'
    = 'REDEFINE' block_a_file_items

block_a_graph_items_redefinition:
    = 'DELETE' 'GRAPH_ITEMS' ';'
    = 'REDEFINE' block_a_graph_items

block_a_grid_items_redefinition:
    = 'DELETE' 'GRID_ITEMS' ';'
    = 'REDEFINE' block_a_grid_items

block_a_image_items_redefinition:
    = 'DELETE' 'IMAGE_ITEMS' ';'
    = 'REDEFINE' block_a_image_items

block_a_item_array_items_redefinition:
    = 'DELETE' 'ITEM_ARRAY_ITEMS' ';'
    = 'REDEFINE' block_a_item_array_items

block_a_list_items_redefinition:
```

```

= 'DELETE' 'LIST_ITEMS' ';'
= 'REDEFINE' block_a_list_items

block_a_local_parameters_redefinition:
= 'LOCAL_PARAMETERS' '{' parameter_redefinition_list '}'

block_a_menu_items_redefinition:
= 'DELETE' 'MENU_ITEMS' ';'
= 'REDEFINE' block_a_menu_items

block_a_method_items_redefinition:
= 'DELETE' 'METHOD_ITEMS' ';'
= 'REDEFINE' block_a_method_items

block_a_parameter_lists_redefinition:
= 'PARAMETER_LISTS' '{' parameter_redefinition_list '}'

block_a_plugin_items_redefinition:
= 'DELETE' 'PLUGIN_ITEMS' ';'
= 'REDEFINE' block_a_plugin_items

block_a_refresh_items_redefinition:
= 'DELETE' 'REFRESH_ITEMS' ';'
= 'REDEFINE' block_a_refresh_items

block_a_source_items_redefinition:
= 'DELETE' 'SOURCE_ITEMS' ';'
= 'REDEFINE' block_a_source_items

block_a_unit_items_redefinition:
= 'DELETE' 'UNIT_ITEMS' ';'
= 'REDEFINE' block_a_unit_items

block_a_wao_items_redefinition:
= 'DELETE' 'WRITE_AS_ONE_ITEMS' ';'
= 'REDEFINE' block_a_wao_items

block_a_waveform_items_redefinition:
= 'DELETE' 'WAVEFORM_ITEMS' ';'
= 'REDEFINE' block_a_waveform_items

block_a_charts_redefinition:
= 'DELETE' 'CHARTS' ';'
= 'REDEFINE' block_a_charts
= 'CHARTS' '{' block_a_member_redefinition_list '}'

block_a_files_redefinition:
= 'DELETE' 'FILES' ';'
= 'REDEFINE' block_a_files
= 'FILES' '{' block_a_member_redefinition_list '}'

block_a_lists_redefinition:
= 'DELETE' 'LISTS' ';'
= 'REDEFINE' block_a_lists
= 'LISTS' '{' block_a_member_redefinition_list '}'

block_a_graphs_redefinition:
= 'DELETE' 'GRAPHS' ';'
= 'REDEFINE' block_a_graphs
= 'GRAPHS' '{' block_a_member_redefinition_list '}'

block_a_grids_redefinition:
= 'DELETE' 'GRIDS' ';'

```

```
= 'REDEFINE' block_a_grids
= 'GRIDS' '{' block_a_member_redefinition_list '}'

block_a_menus_redefinition:
= 'DELETE' 'MENUS' ';'
= 'REDEFINE' block_a_menus
= 'MENUS' '{' block_a_member_redefinition_list '}'

block_a_methods_redefinition:
= 'DELETE' 'METHODS' ';'
= 'REDEFINE' block_a_methods
= 'METHODS' '{' block_a_member_redefinition_list '}'

block_a_plugins_redefinition:
= 'DELETE' 'PLUGINS' ';'
= 'REDEFINE' block_a_plugins
= 'PLUGINS' '{' block_a_member_redefinition_list '}'

block_a_member_redefinition_list:
= block_a_member_redefinition
= block_a_member_redefinition_list block_a_member_redefinition

block_a_member_redefinition:
= 'DELETE' Identifier ';'
= 'REDEFINE' member
= 'ADD' member

block_b_type_redefinition:
= 'REDEFINE' block_b_type

block_b_number_redefinition:
= 'REDEFINE' block_b_number

chart_redefinition:
= 'DELETE' 'CHART' Identifier ';'
= 'REDEFINE' 'CHART' Identifier '{' chart_attribute_list '}'
= 'CHART' Identifier '{' chart_attribute_redefinition_list '}'

chart_attribute_redefinition_list:
= chart_attribute_redefinition
= chart_attribute_redefinition_list chart_attribute_redefinition

chart_attribute_redefinition:
= members_redefinition
= chart_cycle_time_redefinition
= height_redefinition
= help_redefinition
= optional_label_redefinition
= chart_length_redefinition
= chart_type_redefinition
= validity_redefinition
= visibility_redefinition
= width_redefinition

chart_cycle_time_redefinition:
= 'DELETE' 'CYCLE_TIME' ';'
= 'REDEFINE' chart_cycle_time

chart_length_redefinition:
= 'DELETE' 'LENGTH' ';'
= 'REDEFINE' chart_length

chart_type_redefinition:
```

```

= 'DELETE' 'TYPE' ';'
= 'REDEFINE' chart_type

collection_redefinition:
= 'DELETE' 'COLLECTION' Identifier ';'
= 'REDEFINE' 'COLLECTION' Identifier '{' '}'
= 'REDEFINE' 'COLLECTION' 'OF' collection_item_type Identifier '{' '}'
= 'REDEFINE' 'COLLECTION' Identifier '{' collection_attribute_list '}'
= 'REDEFINE' 'COLLECTION' 'OF' collection_item_type Identifier '{'
    collection_attribute_list '}'
= 'COLLECTION' Identifier '{'
    collection_attribute_redefinition_list '}'
= 'COLLECTION' 'OF' collection_item_type Identifier '{'
    collection_attribute_redefinition_list '}'

collection_attribute_redefinition_list:
= collection_attribute_redefinition
= collection_attribute_redefinition_list
    collection_attribute_redefinition

collection_attribute_redefinition:
= members_redefinition
= help_redefinition
= optional_label_redefinition
= validity_redefinition
= visibility_redefinition

command_redefinition:
= 'DELETE' 'COMMAND' Identifier ';'
= 'REDEFINE' 'COMMAND' Identifier '{' '}'
= 'REDEFINE' 'COMMAND' Identifier '{' command_attribute_list '}'
= 'COMMAND' Identifier '{' command_attribute_redefinition_list '}'

command_attribute_redefinition_list:
= command_attribute_redefinition
= command_attribute_redefinition_list command_attribute_redefinition

command_attribute_redefinition:
= command_transaction_redefinition
= command_index_redefinition
= command_slot_redefinition
= command_sub_slot_redefinition
= command_api_redefinition
= command_header_redefinition
= post_rqstreceive_actions_redefinition
= response_codes_reference_redefinition

command_transaction_redefinition:
= 'ADD' command_transaction
= 'REDEFINE' command_transaction
= 'DELETE' 'TRANSACTION' ';'
= 'DELETE' 'TRANSACTION' Integer ';'

command_index_redefinition:
= 'DELETE' 'INDEX' ';'
= 'REDEFINE' command_index

command_slot_redefinition:
= 'DELETE' 'SLOT' ';'
= 'REDEFINE' command_slot

command_sub_slot_redefinition:
= 'DELETE' 'SUB_SLOT' ';'

```

```
= 'REDEFINE' command_sub_slot

command_api_redefinition:
= 'DELETE' 'API' ';'
= 'REDEFINE' command_api

command_header_redefinition:
= 'DELETE' 'HEADER' ';'
= 'REDEFINE' command_header

post_rqstreceive_actions_redefinition:
= 'DELETE' 'POST_RQSTRECEIVE_ACTIONS' ';'
= 'REDEFINE' post_rqstreceive_actions

component_redefinition:
= 'DELETE' 'COMPONENT' Identifier ';'
= 'REDEFINE' 'COMPONENT' Identifier '{' component_attribute_list '}'
= 'COMPONENT' Identifier '{' component_attribute_redefinition_list '}'

component_attribute_redefinition_list:
= component_attribute_redefinition
= component_attribute_redefinition_list
  component_attribute_redefinition

component_attribute_redefinition:
= required_label_redefinition
= byte_order_redefinition
= can_delete_redefinition
= check_configuration_redefinition
= classification_redefinition
= component_parent_redefinition
= component_path_redefinition
= component_relations_redefinition
= connection_point_redefinition
= declaration_redefinition
= detect_method_redefinition
= edd_redefinition
= help_redefinition
= initial_values_redefinition
= product_uri_redefinition
= protocol_redefinition
= redundancy_redefinition
= scan_method_redefinition
= scan_list_redefinition
= supplied_interface_redefinition

byte_order_redefinition:
= 'DELETE' 'BYTE_ORDER' ';'
= 'REDEFINE' byte_order

can_delete_redefinition:
= 'DELETE' 'CAN_DELETE' ';'
= 'REDEFINE' can_delete

check_configuration_redefinition:
= 'DELETE' 'CHECK_CONFIGURATION' ';'
= 'REDEFINE' check_configuration

classification_redefinition:
= 'DELETE' 'CLASSIFICATION' ';'
= 'REDEFINE' classification

component_parent_redefinition:
```

```

= 'DELETE' 'COMPONENT_PARENT' ';'
= 'REDEFINE' component_parent

component_path_redefinition:
= 'DELETE' 'COMPONENT_PATH' ';'
= 'REDEFINE' component_path

component_relations_redefinition:
= 'DELETE' 'COMPONENT_REDEFINITIONS' ';'
= 'REDEFINE' component_relations

connection_point_redefinition:
= 'DELETE' 'CONNECTION_POINT' ';'
= 'REDEFINE' connection_point

declaration_redefinition:
= 'DELETE' 'DECLARATION' ';'
= 'REDEFINE' declaration

detect_method_redefinition:
= 'DELETE' 'DETECT' ';'
= 'REDEFINE' detect_method

edd_redefinition:
= 'DELETE' 'EDD' ';'
= 'REDEFINE' edd

initial_values_redefinition:
= 'DELETE' 'INITIAL_VALUES' ';'
= 'REDEFINE' initial_values

product_uri_redefinition:
= 'DELETE' 'PRODUCT_URI' ';'
= 'REDEFINE' product_uri

protocol_redefinition:
= 'DELETE' 'PROTOCOL' ';'
= 'REDEFINE' protocol

redundancy_redefinition:
= 'DELETE' 'REDUNDANCY' ';'
= 'REDEFINE' redundancy

scan_method_redefinition:
= 'DELETE' 'SCAN' ';'
= 'REDEFINE' scan_method

scan_list_redefinition:
= 'DELETE' 'SCAN_LIST' ';'
= 'REDEFINE' scan_list

supplied_interface_redefinition:
= 'DELETE' 'SUPPLIED_INTERFACE' ';'
= 'REDEFINE' supplied_interface

component_folder_redefinition:
= 'DELETE' 'COMPONENT_FOLDER' Identifier ';'
= 'REDEFINE' 'COMPONENT_FOLDER' Identifier '{'
    component_folder_attribute_list '}'
= 'COMPONENT_FOLDER' Identifier '{'
    component_folder_attribute_redefinition_list '}'

component_folder_attribute_redefinition_list:

```



```
= component_folder_attribute_redefinition
= component_folder_attribute_redefinition_list
  component_folder_attribute_redefinition

component_folder_attribute_redefinition:
= required_label_redefinition
= classification_redefinition
= component_parent_redefinition
= component_path_redefinition
= help_redefinition
= protocol_redefinition

component_reference_redefinition:
= 'DELETE' 'COMPONENT_REFERENCE' Identifier ';'
= 'REDEFINE' 'COMPONENT_REFERENCE' Identifier '{'
  component_reference_attribute_comp_list '}'
= 'COMPONENT_REFERENCE' Identifier '{'
  component_reference_attribute_comp_redefinition_list '}'
= 'REDEFINE' 'COMPONENT_REFERENCE' Identifier '{'
  component_reference_attribute_edd_list '}'
= 'COMPONENT_REFERENCE' Identifier '{'
  component_reference_attribute_edd_redefinition_list '}'

component_reference_attribute_comp_redefinition_list:
= component_reference_attribute_comp_redefinition
= component_reference_attribute_comp_redefinition_list
  component_reference_attribute_comp_redefinition

component_reference_attribute_edd_redefinition_list:
= component_reference_attribute_edd_redefinition
= component_reference_attribute_edd_redefinition_list
  component_reference_attribute_edd_redefinition

component_reference_attribute_comp_redefinition:
= protocol_redefinition
= classification_redefinition
= component_parent_redefinition
= component_path_redefinition

component_reference_attribute_edd_redefinition:
= device_revision_attribute_redefinition
= device_type_attribute_redefinition
= manufacturer_attribute_redefinition

device_revision_attribute_redefinition:
= 'DELETE' 'DEVICE_REVISION' ';'
= 'REDEFINE' device_revision_attribute

device_type_attribute_redefinition:
= 'DELETE' 'DEVICE_TYPE' ';'
= 'REDEFINE' device_type_attribute

manufacturer_attribute_redefinition:
= 'DELETE' 'MANUFACTURER' ';'
= 'REDEFINE' manufacturer_attribute

component_relation_redefinition:
= 'DELETE' 'COMPONENT_RELATION' Identifier ';'
= 'REDEFINE' 'COMPONENT_RELATION' Identifier '{'
  component_relation_attribute_list '}'
= 'COMPONENT_RELATION' Identifier '{'
  component_relation_attribute_redefinition_list '}'
```

```

component_relation_attribute_redefinition_list:
    = component_relation_attribute_redefinition
    = component_relation_attribute_redefinition_list
    component_relation_attribute_redefinition

component_relation_attribute_redefinition:
    = components_redefinition
    = required_label_redefinition
    = relation_type_redefinition
    = addressing_redefinition
    = help_redefinition
    = maximum_number_redefinition
    = minimum_number_redefinition
    = required_interface_redefinition
    = supplied_interface_redefinition

components_redefinition:
    = 'REDEFINE' components
    = 'COMPONENTS' '{' component_relation_reference_redefinition_list '}'

component_relation_reference_redefinition_list:
    = component_relation_reference_redefinition
    = component_relation_reference_redefinition_list ','
    component_relation_reference_redefinition

component_relation_reference_redefinition:
    = 'DELETE' Identifier ';'
    = 'REDEFINE' component_relation_reference_specifier
    = 'ADD' component_relation_reference_specifier
    = Identifier '{' component_relation_arg_redefinition_list '}'

component_relation_arg_redefinition_list:
    = component_relation_arg_redefinition
    = component_relation_arg_redefinition_list
    component_relation_arg_redefinition

component_relation_arg_redefinition:
    = minimum_number_redefinition
    = maximum_number_redefinition
    = auto_create_redefinition
    = filter_redefinition
    = required_ranges_redefinition

auto_create_redefinition:
    = 'DELETE' 'AUTO_CREATE' ';'
    = 'REDEFINE' auto_create

filter_redefinition:
    = 'DELETE' 'FILTER' ';'
    = 'REDEFINE' filter

required_ranges_redefinition:
    = 'DELETE' 'REQUIRED_RANGES' ';'
    = 'REDEFINE' required_ranges

relation_type_redefinition:
    = 'REDEFINE' relation_type

addressing_redefinition:
    = 'DELETE' 'ADDRESSING' ';'
    = 'REDEFINE' addressing

maximum_number_redefinition:

```

```
= 'DELETE' 'MAXIMUM_NUMBER' ';'
= 'REDEFINE' maximum_number

minimum_number_redefinition:
= 'DELETE' 'MINIMUM_NUMBER' ';'
= 'REDEFINE' minimum_number

required_interface_redefinition:
= 'DELETE' 'REQUIRED_INTERFACE' ';'
= 'REDEFINE' required_interface

edit_display_redefinition:
= 'DELETE' 'EDIT_DISPLAY' Identifier ';'
= 'REDEFINE' 'EDIT_DISPLAY' Identifier '{' '}'
= 'REDEFINE' 'EDIT_DISPLAY' Identifier '{'
  edit_display_attribute_list '}'
= 'EDIT_DISPLAY' Identifier '{'
  edit_display_attribute_redefinition_list '}'

edit_display_attribute_redefinition_list:
= edit_display_attribute_redefinition
= edit_display_attribute_redefinition_list
  edit_display_attribute_redefinition

edit_display_attribute_redefinition:
= edit_items_redefinition
= required_label_redefinition
= display_items_redefinition
= post_edit_actions_redefinition
= pre_edit_actions_redefinition

edit_items_redefinition:
= 'REDEFINE' edit_items

display_items_redefinition:
= 'DELETE' 'DISPLAY_ITEMS' ';'
= 'REDEFINE' display_items

file_redefinition:
= 'DELETE' 'FILE' Identifier ';'
= 'REDEFINE' 'FILE' Identifier '{' file_attribute_list '}'
= 'FILE' Identifier '{' file_attribute_redefinition_list '}'

file_attribute_redefinition_list:
= file_attribute_redefinition
= file_attribute_redefinition_list file_attribute_redefinition

file_attribute_redefinition:
= members_redefinition
= optional_label_redefinition
= help_redefinition
= optional_handling_redefinition
= identity_redefinition
= file_shared_redefinition
= on_update_actions_redefinition

file_shared_redefinition:
= 'DELETE' 'SHARED' ';'
= 'REDEFINE' file_shared

on_update_actions_redefinition:
= 'DELETE' 'ON_UPDATE_ACTIONS' ';'
= 'REDEFINE' on_update_actions
```

```

graph_redefinition:
    = 'DELETE' 'GRAPH' Identifier ';'
    = 'REDEFINE' 'GRAPH' Identifier '{' graph_attribute_list '}'
    = 'GRAPH' Identifier '{' graph_attribute_redefinition_list '}'

graph_attribute_redefinition_list:
    = graph_attribute_redefinition
    = graph_attribute_redefinition_list graph_attribute_redefinition

graph_attribute_redefinition:
    = members_redefinition
    = graph_cycle_time_redefinition
    = help_redefinition
    = optional_label_redefinition
    = height_redefinition
    = validity_redefinition
    = visibility_redefinition
    = width_redefinition
    = graph_x_axis_redefinition

graph_cycle_time_redefinition:
    = 'DELETE' 'CYCLE_TIME' ';'
    = 'REDEFINE' graph_cycle_time

graph_x_axis_redefinition:
    = 'DELETE' 'X_AXIS' ';'
    = 'REDEFINE' graph_x_axis

grid_redefinition:
    = 'DELETE' 'GRID' Identifier ';'
    = 'REDEFINE' 'GRID' Identifier '{' grid_attribute_list '}'
    = 'GRID' Identifier '{' grid_attribute_redefinition_list '}'

grid_attribute_redefinition_list:
    = grid_attribute_redefinition
    = grid_attribute_redefinition_list grid_attribute_redefinition

grid_attribute_redefinition:
    = grid_vectors_redefinition
    = optional_handling_redefinition
    = height_redefinition
    = help_redefinition
    = optional_label_redefinition
    = grid_orientation_redefinition
    = validity_redefinition
    = visibility_redefinition
    = width_redefinition

grid_vectors_redefinition:
    = 'REDEFINE' grid_vectors

grid_orientation_redefinition:
    = 'DELETE' 'ORIENTATION' ';'
    = 'REDEFINE' grid_orientation

image_redefinition:
    = 'DELETE' 'IMAGE' Identifier ';'
    = 'REDEFINE' 'IMAGE' Identifier '{' image_attribute_list '}'
    = 'IMAGE' Identifier '{' image_attribute_redefinition_list '}'

image_attribute_redefinition_list:
    = image_attribute_redefinition

```

```

    = image_attribute_redefinition_list image_attribute_redefinition

image_attribute_redefinition:
    = image_path_redefinition
    = optional_label_redefinition
    = help_redefinition
    = image_link_redefinition
    = validity_redefinition
    = visibility_redefinition

image_path_redefinition:
    = 'REDEFINE' image_path

image_link_redefinition:
    = 'DELETE' 'LINK' ';'
    = 'REDEFINE' image_link

interface_redefinition:
    = 'DELETE' 'INTERFACE' Identifier ';'
    = 'REDEFINE' 'INTERFACE' Identifier '{' interface_attribute_list '}'
    = 'INTERFACE' Identifier '{' interface_attribute_redefinition_list '}'

interface_attribute_redefinition_list:
    = interface_attribute_redefinition
    = interface_attribute_redefinition_list
    interface_attribute_redefinition

interface_attribute_redefinition:
    = required_declaration_redefinition
    = required_label_redefinition
    = help_redefinition

required_declaration_redefinition:
    = 'REDEFINE' declaration

item_array_redefinition:
    = 'DELETE' 'ITEM_ARRAY' Identifier ';'
    = 'REDEFINE' 'ITEM_ARRAY' 'OF' collection_item_type Identifier '{' '}'
    = 'REDEFINE' 'ITEM_ARRAY' 'OF' collection_item_type Identifier '{'
        item_array_attribute_list '}'
    = 'ITEM_ARRAY' 'OF' collection_item_type Identifier '{'
        item_array_attribute_redefinition_list '}'

item_array_attribute_redefinition_list:
    = item_array_attribute_redefinition
    = item_array_attribute_redefinition_list
    item_array_attribute_redefinition

item_array_attribute_redefinition:
    = elements_redefinition
    = help_redefinition
    = optional_label_redefinition
    = validity_redefinition

elements_redefinition:
    = 'ELEMENTS' '{' element_redefinition_list '}'
    = 'REDEFINE' elements

element_redefinition_list:
    = element_redefinition
    = element_redefinition_list element_redefinition

element_redefinition:

```

```

= 'DELETE' Integer ';'
= 'REDEFINE' element
= 'ADD' element

list_redefinition:
= 'DELETE' 'LIST' Identifier ';'
= 'REDEFINE' 'LIST' Identifier '{' list_attribute_list '}'
= 'LIST' Identifier '{' list_attribute_redefinition_list '}'

list_attribute_redefinition_list:
= list_attribute_redefinition
= list_attribute_redefinition_list list_attribute_redefinition

list_attribute_redefinition:
= list_type_redefinition
= list_capacity_redefinition
= list_count_redefinition
= help_redefinition
= optional_label_redefinition
= private_redefinition
= validity_redefinition
= visibility_redefinition

list_type_redefinition:
= 'REDEFINE' list_type

list_capacity_redefinition:
= 'DELETE' 'CAPACITY' ';'
= 'REDEFINE' list_capacity

list_count_redefinition:
= 'DELETE' 'COUNT' ';'
= 'REDEFINE' list_count

menu_redefinition:
= 'DELETE' 'MENU' Identifier ';'
= 'REDEFINE' 'MENU' Identifier '{' '}'
= 'REDEFINE' 'MENU' Identifier '{' menu_attribute_list '}'
= 'MENU' Identifier '{' menu_attribute_redefinition_list '}'

menu_attribute_redefinition_list:
= menu_attribute_redefinition
= menu_attribute_redefinition_list menu_attribute_redefinition

menu_attribute_redefinition:
= menu_items_redefinition
= required_label_redefinition
= menu_access_redefinition
= help_redefinition
= menu_exit_actions_redefinition
= menu_init_actions_redefinition
= post_edit_actions_redefinition
= post_read_actions_redefinition
= post_write_actions_redefinition
= pre_edit_actions_redefinition
= pre_read_actions_redefinition
= pre_write_actions_redefinition
= menu_style_redefinition
= validity_redefinition
= visibility_redefinition

menu_items_redefinition:
= 'REDEFINE' menu_items

```

```
menu_access_redefinition:
    = 'DELETE' 'ACCESS' ';'
    = 'REDEFINE' menu_access

menu_style_redefinition:
    = 'DELETE' 'STYLE' ';'
    = 'REDEFINE' menu_style

menu_exit_actions_redefinition:
    = 'DELETE' 'EXIT_ACTIONS' ';'
    = 'REDEFINE' menu_exit_actions

menu_init_actions_redefinition:
    = 'DELETE' 'INIT_ACTIONS' ';'
    = 'REDEFINE' menu_init_actions

method_redefinition:
    = 'DELETE' 'METHOD' Identifier ';'
    = 'REDEFINE' 'METHOD' Identifier '{' '}'
    = 'REDEFINE' 'METHOD' Identifier '{' method_attribute_list '}'
    = 'METHOD' Identifier '{' method_attribute_redefinition_list '}'

method_attribute_redefinition_list:
    = method_attribute_redefinition
    = method_attribute_redefinition_list method_attribute_redefinition

method_attribute_redefinition:
    = method_definition_redefinition
    = method_access_redefinition
    = variable_class_redefinition
    = help_redefinition
    = optional_label_redefinition
    = private_redefinition
    = validity_redefinition
    = visibility_redefinition

method_definition_redefinition:
    = 'REDEFINE' method_definition

method_access_redefinition:
    = 'DELETE' 'ACCESS' ';'
    = 'REDEFINE' method_access

plugin_redefinition:
    = 'DELETE' 'PLUGIN' Identifier ';'
    = 'REDEFINE' 'PLUGIN' Identifier '{' plugin_attribute_list '}'
    = 'PLUGIN' Identifier '{' plugin_attribute_redefinition_list '}'

plugin_attribute_redefinition_list:
    = plugin_attribute_redefinition
    = plugin_attribute_redefinition_list plugin_attribute_redefinition

plugin_attribute_redefinition:
    = help_redefinition
    = plugin_uuid_redefinition
    = required_label_redefinition
    = validity_redefinition
    = visibility_redefinition

plugin_uuid_redefinition:
    = 'REDEFINE' plugin_uuid
```

```

record_redefinition:
    = 'DELETE' 'RECORD' Identifier ';'
    = 'REDEFINE' 'RECORD' Identifier '{' record_attribute_list '}'
    = 'RECORD' Identifier '{' record_attribute_redefinition_list '}'

record_attribute_redefinition_list:
    = record_attribute_redefinition
    = record_attribute_redefinition_list record_attribute_redefinition

record_attribute_redefinition:
    = required_label_redefinition
    = members_redefinition
    = help_redefinition
    = private_redefinition
    = response_codes_reference_redefinition
    = validity_redefinition
    = visibility_redefinition
    = write_mode_redefinition

refresh_relation_redefinition:
    = 'DELETE' 'REFRESH' Identifier ';'
    = 'REDEFINE' 'REFRESH' Identifier '{' '}'
    = 'REDEFINE' 'REFRESH' Identifier
      '{' refresh_relation_cause ':' refresh_relation_effect '}'

unit_relation_redefinition:
    = 'DELETE' 'UNIT' Identifier ';'
    = 'REDEFINE' 'UNIT' Identifier '{' '}'
    = 'REDEFINE' 'UNIT' Identifier
      '{' unit_relation_cause ':' unit_relation_effect '}'

wao_relation_redefinition:
    = 'DELETE' 'WRITE_AS_ONE' Identifier ';'
    = 'REDEFINE' 'WRITE_AS_ONE' Identifier '{' '}'
    = 'REDEFINE' 'WRITE_AS_ONE' Identifier
      '{' wao_specifier '}'

response_codes_definition_redefinition:
    = 'DELETE' 'RESPONSE_CODES' Identifier ';'
    = 'REDEFINE' 'RESPONSE_CODES' Identifier '{' '}'
    = 'REDEFINE' 'RESPONSE_CODES' Identifier '{'
      response_codes_specifier_list '}'
    = 'RESPONSE_CODES' Identifier '{' response_code_redefinition_list
      '}'

response_code_redefinition_list:
    = response_code_redefinition
    = response_code_redefinition_list response_code_redefinition

response_code_redefinition:
    = 'DELETE' Integer ';'
    = 'REDEFINE' response_code
    = 'ADD' response_code

emphasis_redefinition:
    = 'DELETE' 'EMPHASIS' ';'
    = 'REDEFINE' emphasis

height_redefinition:
    = 'DELETE' 'HEIGHT' ';'
    = 'REDEFINE' height

help_redefinition:

```



```
= 'DELETE' 'HELP' ';'
= 'REDEFINE' help

required_label_redefinition:
= 'REDEFINE' label

optional_label_redefinition:
= 'DELETE' 'LABEL' ';'
= 'REDEFINE' label

line_color_redefinition:
= 'DELETE' 'LINE_COLOR' ';'
= 'REDEFINE' line_color

line_type_redefinition:
= 'DELETE' 'LINE_TYPE' ';'
= 'REDEFINE' line_type

response_codes_reference_redefinition:
= 'DELETE' 'RESPONSE_CODES' ';'
= 'REDEFINE' response_codes

private_redefinition:
= 'DELETE' 'PRIVATE' ';'
= 'REDEFINE' private

validity_redefinition:
= 'DELETE' 'VALIDITY' ';'
= 'REDEFINE' validity

visibility_redefinition:
= 'DELETE' 'VISIBILITY' ';'
= 'REDEFINE' visibility

width_redefinition:
= 'DELETE' 'WIDTH' ';'
= 'REDEFINE' width

write_mode_redefinition:
= 'DELETE' 'WRITE_MODE' ';'
= 'REDEFINE' write_mode

members_redefinition:
= 'MEMBERS' '{' member_redefinition_list '}'
= 'REDEFINE' members

member_redefinition_list:
= member_redefinition
= member_redefinition_list member_redefinition

member_redefinition:
= 'DELETE' Identifier ';'
= 'REDEFINE' member
= 'ADD' member

source_redefinition:
= 'DELETE' 'SOURCE' Identifier ';'
= 'REDEFINE' 'SOURCE' Identifier '{' source_attribute_list '}'
= 'SOURCE' Identifier '{' source_attribute_redefinition_list '}'

source_attribute_redefinition_list:
= source_attribute_redefinition
= source_attribute_redefinition_list source_attribute_redefinition
```

```

source_attribute_redefinition:
    = members_redefinition
    = emphasis_redefinition
    = source_exit_actions_redefinition
    = help_redefinition
    = source_init_actions_redefinition
    = optional_label_redefinition
    = line_color_redefinition
    = line_type_redefinition
    = source_refresh_actions_redefinition
    = validity_redefinition
    = source_y_axis_redefinition

source_exit_actions_redefinition:
    = 'DELETE' 'EXIT_ACTIONS' ';'
    = 'REDEFINE' source_exit_actions

source_init_actions_redefinition:
    = 'DELETE' 'INIT_ACTIONS' ';'
    = 'REDEFINE' source_init_actions

source_refresh_actions_redefinition:
    = 'DELETE' 'REFRESH_ACTIONS' ';'
    = 'REDEFINE' source_refresh_actions

source_y_axis_redefinition:
    = 'DELETE' 'Y_AXIS' ';'
    = 'REDEFINE' waveform_y_axis

template_redefinition:
    = 'DELETE' 'TEMPLATE' Identifier ';'
    = 'REDEFINE' 'TEMPLATE' Identifier '{' template_attribute_list '}'
    = 'TEMPLATE' Identifier '{' template_attribute_redefinition_list '}'

template_attribute_redefinition_list:
    = template_attribute_redefinition
    = template_attribute_redefinition_list template_attribute_redefinition

template_attribute_redefinition:
    = template_default_values_redefinition
    = required_label_redefinition
    = help_redefinition
    = validity_redefinition

template_default_values_redefinition:
    = 'REDEFINE' template_default_values

variable_redefinition:
    = 'DELETE' 'VARIABLE' Identifier ';'
    = 'REDEFINE' 'VARIABLE' Identifier '{' variable_attribute_list
    '}'
    = 'VARIABLE' Identifier '{' variable_attribute_redefinition_list
    '}'

variable_attribute_redefinition_list:
    = variable_attribute_redefinition
    = variable_attribute_redefinition_list variable_attribute_redefinition

variable_attribute_redefinition:
    = variable_class_redefinition
    = type_redefinition
    = constant_unit_redefinition

```

```

= default_value_redefinition
= optional_handling_redefinition
= height_redefinition
= help_redefinition
= initial_value_redefinition
= optional_label_redefinition
= post_edit_actions_redefinition
= post_read_actions_redefinition
= post_rqstupdate_actions_redefinition
= post_userchange_actions_redefinition
= post_write_actions_redefinition
= pre_edit_actions_redefinition
= pre_read_actions_redefinition
= pre_write_actions_redefinition
= private_redefinition
= refresh_actions_redefinition
= response_codes_reference_redefinition
= validity_redefinition
= visibility_redefinition
= width_redefinition
= write_mode_redefinition

variable_class_redefinition:
= 'REDEFINE' variable_class

type_redefinition:
= 'REDEFINE' 'TYPE' arithmetic_type
= 'TYPE' arithmetic_type_redefinition
= 'REDEFINE' 'TYPE' enumerated_type
= 'TYPE' enumerated_type_redefinition
= 'REDEFINE' 'TYPE' string_type
= 'TYPE' string_type_redefinition
= 'REDEFINE' 'TYPE' date_time_type
= 'TYPE' date_time_type_redefinition

arithmetic_type_redefinition:
= 'INTEGER' arithmetic_size '{'
  arithmetic_option_redefinition_list '}'
= 'UNSIGNED_INTEGER' arithmetic_size '{'
  arithmetic_option_redefinition_list '}'
= 'FLOAT' '{' arithmetic_option_redefinition_list '}'
= 'DOUBLE' '{' arithmetic_option_redefinition_list '}'

arithmetic_option_redefinition_list:
= arithmetic_option_redefinition
= arithmetic_option_redefinition_list arithmetic_option_redefinition

arithmetic_option_redefinition:
= default_value_redefinition
= initial_value_redefinition
= display_format_redefinition
= edit_format_redefinition
= arithmetic_enumerator_list_redefinition
= maximum_value_redefinition
= maximum_value_n_redefinition
= minimum_value_redefinition
= minimum_value_n_redefinition
= scaling_factor_redefinition

display_format_redefinition:
= 'DELETE' 'DISPLAY_FORMAT' ';'
= 'REDEFINE' display_format

```

```

edit_format_redefinition:
    = 'DELETE' 'EDIT_FORMAT' ';'
    = 'REDEFINE' edit_format

arithmetic_enumerator_list_redefinition:
    = 'DELETE' Integer ';'
    = 'DELETE' Real ';'
    = 'REDEFINE' arithmetic_enumerator
    = 'ADD' arithmetic_enumerator

maximum_value_redefinition:
    = 'DELETE' 'MAX_VALUE' ';'
    = 'REDEFINE' maximum_value

maximum_value_n_redefinition:
    = 'DELETE' 'MAX_VALUE1' ';'
    = 'DELETE' 'MAX_VALUE2' ';'
    = 'DELETE' 'MAX_VALUE3' ';'
    = 'DELETE' 'MAX_VALUE4' ';'
    = 'DELETE' 'MAX_VALUE5' ';'
    = 'DELETE' 'MAX_VALUE6' ';'
    = 'DELETE' 'MAX_VALUE7' ';'
    = 'DELETE' 'MAX_VALUE8' ';'
    = 'DELETE' 'MAX_VALUE9' ';'
    = 'REDEFINE' maximum_value_n

minimum_value_redefinition:
    = 'DELETE' 'MIN_VALUE' ';'
    = 'REDEFINE' minimum_value

minimum_value_n_redefinition:
    = 'DELETE' 'MIN_VALUE1' ';'
    = 'DELETE' 'MIN_VALUE2' ';'
    = 'DELETE' 'MIN_VALUE3' ';'
    = 'DELETE' 'MIN_VALUE4' ';'
    = 'DELETE' 'MIN_VALUE5' ';'
    = 'DELETE' 'MIN_VALUE6' ';'
    = 'DELETE' 'MIN_VALUE7' ';'
    = 'DELETE' 'MIN_VALUE8' ';'
    = 'DELETE' 'MIN_VALUE9' ';'
    = 'REDEFINE' minimum_value_n

scaling_factor_redefinition:
    = 'DELETE' 'SCALING_FACTOR' ';'
    = 'REDEFINE' scaling_factor

enumerated_type_redefinition:
    = 'ENUMERATED' enumerated_size '{'
        enumerated_option_redefinition_list '}'
    = 'BIT_ENUMERATED' enumerated_size '{'
        bit_enumerated_option_redefinition_list '}'

enumerated_option_redefinition_list:
    = enumerated_option_redefinition
    = enumerated_option_redefinition_list enumerated_option_redefinition

enumerated_option_redefinition:
    = default_value_redefinition
    = initial_value_redefinition
    = integral_enumerator_redefinition

integral_enumerator_redefinition:
    = 'DELETE' Integer ';'

```

```

    = 'REDEFINE' integral_enumerator
    = 'ADD' integral_enumerator

bit_enumerated_option_redefinition_list:
    = bit_enumerated_option_redefinition
    = bit_enumerated_option_redefinition_list
bit_enumerated_option_redefinition

bit_enumerated_option_redefinition:
    = default_value_redefinition
    = initial_value_redefinition
    = bit_enumerators_redefinition

bit_enumerators_redefinition:
    = 'DELETE' Integer ';'
    = 'REDEFINE' bit_enumerator
    = 'ADD' bit_enumerator

string_type_redefinition:
    = 'ASCII' string_size '{' string_option_redefinition_list '}'
    = 'BITSTRING' string_size '{' string_option_redefinition_list '}'
    = 'EUC' string_size '{' string_option_redefinition_list '}'
    = 'OCTET' string_size '{' string_option_redefinition_list '}'
    = 'PACKED_ASCII' string_size '{' string_option_redefinition_list '}'
    = 'PASSWORD' string_size '{' string_option_redefinition_list '}'
    = 'VISIBLE' string_size '{' string_option_redefinition_list '}'

string_option_redefinition_list:
    = string_option_redefinition
    = string_option_redefinition_list string_option_redefinition

string_option_redefinition:
    = string_default_value_redefinition
    = string_initial_value_redefinition
    = string_enumerator_redefinition

string_default_value_redefinition:
    = 'DELETE' 'DEFAULT_VALUE' ';'
    = 'REDEFINE' string_default_value

string_initial_value_redefinition:
    = 'DELETE' 'INITIAL_VALUE' ';'
    = 'REDEFINE' string_initial_value

string_enumerator_redefinition:
    = 'DELETE' string ';'
    = 'REDEFINE' string_enumerator
    = 'ADD' string_enumerator

date_time_type_redefinition:
    = 'DATE' '{' date_time_option_redefinition_list '}'
    = 'DATE AND TIME' '{' date_time_option_redefinition_list '}'
    = 'DURATION' '{' date_time_option_redefinition_list '}'
    = 'TIME' '{' date_time_option_redefinition_list '}'
    = 'TIME_VALUE' date_time_size '{'
        = time_value_option_redefinition_list '}'

date_time_option_redefinition_list:
    = date_time_option_redefinition
    = date_time_option_redefinition_list date_time_option_redefinition

date_time_option_redefinition:
    = default_value_redefinition

```

```

    = initial_value_redefinition

time_value_option_redefinition_list:
    = time_value_option_redefinition
    = time_value_option_redefinition_list time_value_option_redefinition

time_value_option_redefinition:
    = default_value_redefinition
    = display_format_redefinition
    = edit_format_redefinition
    = initial_value_redefinition
    = time_format_redefinition
    = time_scale_redefinition

time_format_redefinition:
    = 'DELETE' 'TIME_FORMAT' ';'
    = 'REDEFINE' time_format

time_scale_redefinition:
    = 'DELETE' 'TIME_SCALE' ';'
    = 'REDEFINE' time_scale

constant_unit_redefinition:
    = 'DELETE' 'CONSTANT_UNIT' ';'
    = 'REDEFINE' constant_unit

default_value_redefinition:
    = 'DELETE' 'DEFAULT_VALUE' ';'
    = 'REDEFINE' default_value

required_handling_redefinition:
    = 'REDEFINE' handling

optional_handling_redefinition:
    = 'DELETE' 'HANDLING' ';'
    = 'REDEFINE' handling

initial_value_redefinition:
    = 'DELETE' 'INITIAL_VALUE' ';'
    = 'REDEFINE' initial_value

post_edit_actions_redefinition:
    = 'DELETE' 'POST_EDIT_ACTIONS' ';'
    = 'REDEFINE' post_edit_actions

post_rqstupdate_actions_redefinition:
    = 'DELETE' 'POST_RQSTUPDATE_ACTIONS' ';'
    = 'REDEFINE' post_rqstupdate_actions

post_userchange_actions_redefinition:
    = 'DELETE' 'POST_USERCHANGE_ACTIONS' ';'
    = 'REDEFINE' post_userchange_actions

post_read_actions_redefinition:
    = 'DELETE' 'POST_READ_ACTIONS' ';'
    = 'REDEFINE' post_read_actions

post_write_actions_redefinition:
    = 'DELETE' 'POST_WRITE_ACTIONS' ';'
    = 'REDEFINE' post_write_actions

pre_edit_actions_redefinition:
    = 'DELETE' 'PRE_EDIT_ACTIONS' ';'

```

```

    = 'REDEFINE' pre_edit_actions

pre_read_actions_redefinition:
    = 'DELETE' 'PRE_READ_ACTIONS' ';'
    = 'REDEFINE' pre_read_actions

pre_write_actions_redefinition:
    = 'DELETE' 'PRE_WRITE_ACTIONS' ';'
    = 'REDEFINE' pre_write_actions

refresh_actions_redefinition:
    = 'DELETE' 'REFRESH_ACTIONS' ';'
    = 'REDEFINE' refresh_actions

variable_list_redefinition:
    = 'DELETE' 'VARIABLE_LIST' Identifier ';'
    = 'REDEFINE' 'VARIABLE_LIST' Identifier '{' '}'
    = 'REDEFINE' 'VARIABLE_LIST' Identifier '{'
        variable_list_attribute_list '}'
    = 'VARIABLE_LIST' Identifier '{'
        variable_list_attribute_redefinition_list '}'

variable_list_attribute_redefinition_list:
    = variable_list_attribute_redefinition
    = variable_list_attribute_redefinition_list
        variable_list_attribute_redefinition

variable_list_attribute_redefinition:
    = members_redefinition
    = help_redefinition
    = optional_label_redefinition
    = response_codes_reference_redefinition

waveform_redefinition:
    = 'DELETE' 'WAVEFORM' Identifier ';'
    = 'REDEFINE' 'WAVEFORM' Identifier '{' waveform_attribute_list '}'
    = 'WAVEFORM' Identifier '{' waveform_attribute_redefinition_list '}'

waveform_attribute_redefinition_list:
    = waveform_attribute_redefinition
    = waveform_attribute_redefinition_list waveform_attribute_redefinition

waveform_attribute_redefinition:
    = waveform_type_redefinition
    = emphasis_redefinition
    = waveform_exit_actions_redefinition
    = optional_handling_redefinition
    = help_redefinition
    = waveform_init_actions_redefinition
    = waveform_key_points_redefinition
    = optional_label_redefinition
    = line_color_redefinition
    = line_type_redefinition
    = waveform_refresh_actions_redefinition
    = validity_redefinition
    = waveform_y_axis_redefinition

waveform_type_redefinition:
    = 'REDEFINE' waveform_type
    = 'TYPE' waveform_type_option_redefinition

waveform_type_option_redefinition:
    = 'YT' '{' waveform_yt_type_option_redefinition_list '}'

```

```

= 'XY' '{' waveform_xy_type_option_redefinition_list '}'
= 'HORIZONTAL' '{' waveform_y_values_redefinition '}'
= 'VERTICAL' '{' waveform_x_values_redefinition '}'

waveform_yt_type_option_redefinition_list:
= waveform_yt_type_option_redefinition
= waveform_yt_type_option_redefinition_list
waveform_yt_type_option_redefinition

waveform_yt_type_option_redefinition:
= yt_x_initial_redefinition
= yt_x_increment_redefinition
= waveform_y_values_redefinition
= number_of_points_redefinition

yt_x_initial_redefinition:
= 'REDEFINE' yt_x_initial

yt_x_increment_redefinition:
= 'REDEFINE' yt_x_increment

waveform_y_values_redefinition:
= 'REDEFINE' waveform_y_values
= 'ADD' waveform_y_values

waveform_x_values_redefinition:
= 'REDEFINE' waveform_x_values
= 'ADD' waveform_x_values

number_of_points_redefinition:
= 'DELETE' 'NUMBER_OF_POINTS' ';'
= 'REDEFINE' number_of_points

waveform_xy_type_option_redefinition_list:
= waveform_xy_type_option_redefinition
= waveform_xy_type_option_redefinition_list
waveform_xy_type_option_redefinition

waveform_xy_type_option_redefinition:
= waveform_x_values_redefinition
= waveform_y_values_redefinition
= number_of_points_redefinition

waveform_exit_actions_redefinition:
= 'DELETE' 'EXIT_ACTIONS' ';'
= 'REDEFINE' waveform_exit_actions

waveform_init_actions_redefinition:
= 'DELETE' 'INIT_ACTIONS' ';'
= 'REDEFINE' waveform_init_actions

waveform_key_points_redefinition:
= 'DELETE' 'KEY_POINTS' ';'
= 'REDEFINE' 'KEY_POINTS' '{' key_points_options_redefinition_list '}'

key_points_options_redefinition_list:
= key_points_options_redefinition
= key_points_options_redefinition_list key_points_options_redefinition

key_points_options_redefinition:
= waveform_x_values_redefinition
= waveform_y_values_redefinition

```



```

waveform_refresh_actions_redefinition:
    = 'DELETE' 'REFRESH_ACTIONS' ';'
    = 'REDEFINE' waveform_refresh_actions

```

```

waveform_y_axis_redefinition:
    = 'DELETE' 'Y_AXIS' ';'
    = 'REDEFINE' waveform_y_axis

```

A.6.40 References

```

reference:
    = reference_without_call
    = reference_without_call '(' argument_list ')'

```

```

reference_without_call:
    = Identifier
    = reference_without_call '[' expr ']'
    = reference_without_call '.' Identifier
    = reference_without_call '.' selector
    = reference_without_call '->' Identifier
    = 'BLOCK' '.' Identifier
    = 'LOCAL_PARAM' '.' Identifier
    = 'PARAM' '.' Identifier
    = 'PARAM_LIST' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'PARAM' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'LOCAL_PARAM' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'BLOCK' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'CHART' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'LIST' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'GRAPH' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'GRID' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'MENU' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'METHOD' '.' Identifier
    = block_a_reference '[' expr ']' '.' 'PLUGIN' '.' Identifier

```

```

argument_list:
    = argument
    = argument_list ',' argument

```

```

argument:
    = expr

```

```

selector:
    = string_selector
    = 'CAPACITY'
    = 'COUNT'
    = 'DEFAULT_VALUE'
    = 'FIRST'
    = 'INITIAL_VALUE'
    = 'LAST'
    = 'MAX_VALUE'
    = 'MAX_VALUE1'
    = 'MAX_VALUE2'
    = 'MAX_VALUE3'
    = 'MAX_VALUE4'
    = 'MAX_VALUE5'
    = 'MAX_VALUE6'
    = 'MAX_VALUE7'
    = 'MAX_VALUE8'
    = 'MAX_VALUE9'
    = 'MIN_VALUE'
    = 'MIN_VALUE1'
    = 'MIN_VALUE2'
    = 'MIN_VALUE3'

```

```

= 'MIN_VALUE4'
= 'MIN_VALUE5'
= 'MIN_VALUE6'
= 'MIN_VALUE7'
= 'MIN_VALUE8'
= 'MIN_VALUE9'
= 'NUMBER_OF_ELEMENTS'
= 'SCALING'
= 'SCALING_FACTOR'
= 'VARIABLE_STATUS'
= 'VIEW_MAX'
= 'VIEW_MIN'
= 'X_AXIS'
= 'Y_AXIS'

string_selector:
= 'CONSTANT_UNIT'
= 'HELP'
= 'LABEL'
= 'IDENTITY'

dictionary_reference:
= '[' Identifier ']'

string_attribute_reference:
= reference_without_call '.' string_selector

array_reference:
= reference_without_call

axis_reference:
= reference_without_call

blob_reference:
= reference_without_call

block_a_reference:
= reference_without_call

block_b_reference:
= reference_without_call

chart_reference:
= reference_without_call

collection_reference:
= reference_without_call

component_folder_reference:
= reference_without_call

component_relation_reference:
= reference_without_call

edit_display_reference:
= reference_without_call

file_reference:
= reference_without_call

graph_reference:
= reference_without_call

```

```
grid_reference:
    = reference_without_call

image_reference:
    = reference_without_call

interface_reference:
    = reference_without_call

list_reference:
    = reference_without_call

item_array_reference:
    = reference_without_call

menu_reference:
    = reference_without_call

method_reference:
    = reference

plugin_reference:
    = reference_without_call

record_reference:
    = reference_without_call

refresh_reference:
    = reference_without_call

response_code_reference:
    = reference_without_call

source_reference:
    = reference_without_call

unit_reference:
    = reference_without_call

variable_reference:
    = reference_without_call

enumerated variable reference:
    = variable_reference

wao_reference:
    = reference_without_call

waveform_reference:
    = reference_without_call
```

A.6.41 PLUGIN

```
plugin:
    = 'PLUGIN' Identifier '{' plugin_attribute_list '}'

plugin_attribute_list:
    = plugin_attribute
    = plugin_attribute_list plugin_attribute

plugin_attribute:
    = label
    = help
```

/* M */
/* O */

```

= uuid                /* M */
= validity            /* O */
= visibility          /* O */

```

```

plugin_uuid:
  = 'UUID' uuid ';'

```

A.6.42 BLOB

```

blob:
  = 'BLOB' Identifier '{' blob_attribute_list '}'

```

```

blob_attribute_list:
  = blob_attribute
  = blob_attribute_list blob_attribute

```

```

blob_attribute:
  = handling          /* M */
  = help              /* O */
  = identity          /* O */
  = label             /* O */

```

A.7 Formal dictionary syntax

The dictionary shall not be preprocessed by the C-preprocessor but comments are allowed as defined in A.1.5.

```

dictionary_description:
  = dictionary_entry_list

dictionary_entry_list:
  = dictionary_entry
  = dictionary_entry_list dictionary_entry

dictionary_entry:
  = '[' section_number ',' string_number ']' Identifier string_literal

section_number:
  = Integer

string_number:
  = Integer

/*
  Already defined in A.5: Integer, Identifier
  Already defined in A.6: string_literal
*/

```

Annex B
(normative)

EDDL Builtin library (void)

The EDDL Builtin library is specified in IEC 61804-5.

IECNORM.COM: Click to view the full PDF of IEC 61804-3:2015

Withd2W

Annex C (informative)

EDD example

C.1 EDD example of a temperature transmitter

The following EDD example describes a temperature transmitter which is composed of one Device Block, one Analog Input Function Block and one Temperature Technology Block. The general structure of the EDD file is as follows.

- The start is the EDD identification with the keywords according to 7.2.
- There are the general sections for one Temperature Technology Block, one Analog Input Function Block and one Device Block. Each section starts with the block description (block_b type) which is followed by its parameters.
- Each parameter is described by its variable with attributes and its command describing the communication transactions.
- The menu section of the description describes the structure of the operator interface. The design of the screen is tool specific.

Parameter	Value	Unit	State	Name in DD
Main_MENU				Table_Main_Specialist
>> Menu_Operator				Tab_s_Operator
DEVICE_ID	IEC61804 TEMP TR		initial	DeviceBlock_DEVICE_ID
OUT_Value	0.000		valid	AnalogInputFB_OUT_Value
OUT_Status	good		valid	AnalogInputFB_OUT_Status
OUT_SCALE_UnitsIndex	degree_Celsius		initial	AnalogInputFB_OUT_SCALE_UnitsIndex
>> Menu_Specialist				Tab_s_Specialist
>>> Menu_AnalogInputFunctionBlock				Tab_s_AnalogInputFunctionBlock
OUT_Value	0.000		valid	AnalogInputFB_OUT_Value
OUT_Status	good		valid	AnalogInputFB_OUT_Status
OUT_SCALE_EUat100percent	100.000		initial	AnalogInputFB_OUT_SCALE_EUat100percent
OUT_SCALE_EUat0percent	0.000		initial	AnalogInputFB_OUT_SCALE_EUat0percent
OUT_SCALE_UnitsIndex	degree_Celsius		initial	AnalogInputFB_OUT_SCALE_UnitsIndex
OUT_SCALE_DecimalPoint	0		valid	AnalogInputFB_OUT_SCALE_DecimalPoint
HI_LIM	3.400000e+038		initial	AnalogInputFB_HI_LIM
LO_LIM	-3.400000e+038		initial	AnalogInputFB_LO_LIM
>>> Menu_TemperatureTechnologyBlock				Tab_s_TemperatureTechnologyBlock
PRIMARY_VALUE_Value	0.000		valid	TemperatureTB_PRIMARY_VALUE_Value
PRIMARY_VALUE_Status	good		valid	TemperatureTB_PRIMARY_VALUE_Status
PRIMARY_VALUE_UNIT	degree_Celsius		initial	TemperatureTB_PRIMARY_VALUE_UNIT
SECONDARY_VALUE_1_Value	0.000		valid	TemperatureTB_SECONDARY_VALUE_1_Value
SECONDARY_VALUE_1_Status	good		valid	TemperatureTB_SECONDARY_VALUE_1_Status
SECONDARY_VALUE_2_Value	0.000		valid	TemperatureTB_SECONDARY_VALUE_2_Value
SECONDARY_VALUE_2_Status	good		valid	TemperatureTB_SECONDARY_VALUE_2_Status
LIN_TYPE	Pbxxx		valid	TemperatureTB_LIN_TYPE
SENSOR_CONNECTION	4 wires		valid	TemperatureTB_SENSOR_CONNECTION
>>> Menu_DeviceBlock				Tab_s_DeviceBlock
DEVICE_MAN_ID	0		valid	DeviceBlock_DEVICE_MAN_ID
DEVICE_ID	IEC61804 TEMP TR		initial	DeviceBlock_DEVICE_ID
DEVICE_SER_Num	chdlwelskfrgwtb		initial	DeviceBlock_DEVICE_SER_Num

Figure C.1 – Example of an operator screen using EDD

Figure C.1 shows an example of how the temperature transmitter can be visualised at an operator screen. On the left side there is the menu structure (Main_MENU, Menu_Operator, Menu_Specialist). The Menu_Operator references the measurement value as well as its unit and status. The Menu_Specialist has the submenus Analog_Input_Function_Block, Temperature_Technology_Block and Device_Block. On the right-hand side, there is a table-oriented view of the menu details, i.e. the parameters which are referenced in the relevant menu definition. This representation is based on the EDD example in Clause C.2.

C.2 EDD example

```

/* -----
 * Copyright (C)   IEC 61804
 * Product:       EDD
 * Device:        IEC 61804 Temperature Transmitter
 * Description:    Electronic Device Description (EDD)
 * Dictionary:
 * $Revision:      Name
 * $Date:          2005-07-17
 * -----*/

MANUFACTURER      0,
DEVICE_TYPE       0,
DEVICE_REVISION   1,
DD_REVISION       1

/* -----
 * DEFINE
 * -----*/

#define DEF_STATUS \
    { 0x00 , [bad]      }, \
    { 0x40 , [uncertain] }, \
    { 0x80 , [good]     }

/* -----
 * Device Block
 * -----*/

BLOCK DeviceBlock
{
    TYPE PHYSICAL;
    NUMBER 1;
}

//----DB-10----

VARIABLE DeviceBlock__DEVICE_MAN_ID
{
    LABEL [DEVICE_MAN_ID];
    HELP [DeviceBlock__DEVICE_MAN_ID_Help];
    CLASS CONTAINED;
    TYPE UNSIGNED_INTEGER (2);
    HANDLING READ;
}

COMMAND Read__DeviceBlock__DEVICE_MAN_ID
{
    BLOCK DeviceBlock;
    INDEX 10;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            DeviceBlock__DEVICE_MAN_ID
        }
    }
}

```

//----DB-11----

```
VARIABLE DeviceBlock__DEVICE_ID
{
    LABEL [DEVICE_ID];
    HELP [DeviceBlock__DEVICE_ID_Help];
    CLASS CONTAINED;
    TYPE ASCII (16)
    {
        DEFAULT_VALUE "IEC61804 TEMP TR";
    }
    HANDLING READ;
}
```

COMMAND Read__DeviceBlock__DEVICE_ID

```
{
    BLOCK DeviceBlock;
    INDEX 11;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            DeviceBlock__DEVICE_ID
        }
    }
}
```

//----DB-12----

```
VARIABLE DeviceBlock__DEVICE_SER_Num
{
    LABEL [DEVICE_SER_Num];
    HELP [DeviceBlock__DEVICE_SER_Num_Help];
    CLASS CONTAINED;
    TYPE ASCII (16)
    {
        DEFAULT_VALUE "chdelskfrgwlwtb";
    }
    HANDLING READ;
}
```

COMMAND Read__DeviceBlock__DEVICE_SER_Num

```
{
    BLOCK DeviceBlock;
    INDEX 12;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            DeviceBlock__DEVICE_SER_Num
        }
    }
}
```



```

/* -----
 * Analog Input Function Block
 * -----*/

```

```

BLOCK AnalogInputFB

```

```

{
    TYPE FUNCTION;
    NUMBER 1;
}

```

```

//----AIFB-10----

```

```

VARIABLE AnalogInputFB__OUT__Value

```

```

{
    LABEL [OUT__Value];
    HELP [AnalogInputFB__OUT__Help];
    CLASS CONTAINED;
    TYPE FLOAT;
    HANDLING READ;
}

```

```

VARIABLE AnalogInputFB__OUT__Status

```

```

{
    LABEL [OUT__Status];
    HELP [AnalogInputFB__OUT__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (1)
    {
        DEF_STATUS
    }
    HANDLING READ;
}

```

```

COMMAND Read__AnalogInputFB__OUT

```

```

{
    BLOCK AnalogInputFB;
    INDEX 10;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            AnalogInputFB__OUT__Value,
            AnalogInputFB__OUT__Status
        }
    }
}

```

```

//----AIFB-12----

```

```

VARIABLE AnalogInputFB__OUT__SCALE__EUat100percent

```

```

{
    LABEL [OUT__SCALE__EUat100percent];
    HELP [AnalogInputFB__OUT__SCALE__EUat100percent__Help];
    CLASS CONTAINED;
    TYPE FLOAT
    {
        DEFAULT_VALUE 100;
    }
    HANDLING READ & WRITE;
}

```

```

}

VARIABLE AnalogInputFB__OUT_SCALE__EUat0percent
{
    LABEL [OUT_SCALE__EUat0percent];
    HELP [AnalogInputFB__OUT_SCALE__EUat0percent__Help];
    CLASS CONTAINED;
    TYPE FLOAT
    {
        DEFAULT_VALUE 0;
    }
    HANDLING READ & WRITE;
}

VARIABLE AnalogInputFB__OUT_SCALE__UnitsIndex
{
    LABEL [OUT_SCALE__UnitsIndex];
    HELP [AnalogInputFB__OUT_SCALE__UnitsIndex__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (2)
    {
        DEFAULT_VALUE 1001;
        { 1000 , [Kelvin], [Kelvin_help] },
        { 1001 , [degree_Celsius], [degree_Celsius_help] },
        { 1002 , [degree_Fahrenheit], [degree_Fahrenheit_help] },
        { 1003 , [degree_Rankine], [degree_Rankine_help] },
        { 1243 , [millivolt], [millivolt_help] },
        { 1281 , [Ohm], [Ohm_help] }
    }
    HANDLING READ & WRITE;
}

VARIABLE AnalogInputFB__OUT_SCALE__DecimalPoint
{
    LABEL [OUT_SCALE__DecimalPoint];
    HELP [AnalogInputFB__OUT_SCALE__DecimalPoint__Help];
    CLASS CONTAINED;
    TYPE INTEGER (1);
    HANDLING READ & WRITE;
}

COMMAND Read__AnalogInputFB__OUT_SCALE
{
    BLOCK AnalogInputFB;
    INDEX 12;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            AnalogInputFB__OUT_SCALE__EUat100percent,
            AnalogInputFB__OUT_SCALE__EUat0percent,
            AnalogInputFB__OUT_SCALE__UnitsIndex,
            AnalogInputFB__OUT_SCALE__DecimalPoint
        }
    }
}

COMMAND Write__AnalogInputFB__OUT_SCALE
{

```

```

BLOCK AnalogInputFB;
INDEX 12;
OPERATION WRITE;
TRANSACTION
{
    REQUEST
    {
        AnalogInputFB__OUT_SCALE__EUat100percent,
        AnalogInputFB__OUT_SCALE__EUat0percent,
        AnalogInputFB__OUT_SCALE__UnitsIndex,
        AnalogInputFB__OUT_SCALE__DecimalPoint
    }
    REPLY
    {
    }
}

//----AIFB-23----

VARIABLE AnalogInputFB__HI_LIM
{
    LABEL [HI_LIM];
    HELP [AnalogInputFB__HI_LIM__Help];
    CLASS CONTAINED;
    TYPE FLOAT
    {
        DEFAULT_VALUE 3.4E+38;
    }
    HANDLING READ & WRITE;
}

COMMAND Read__AnalogInputFB__HI_LIM
{
    BLOCK AnalogInputFB;
    INDEX 23;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            AnalogInputFB__HI_LIM
        }
    }
}

COMMAND Write__AnalogInputFB__HI_LIM
{
    BLOCK AnalogInputFB;
    INDEX 23;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            AnalogInputFB__HI_LIM
        }
        REPLY
        {
        }
    }
}

```

```

    }
}

//----AIFB-25----

VARIABLE AnalogInputFB__LO_LIM
{
    LABEL [LO_LIM];
    HELP [AnalogInputFB__LO_LIM__Help];
    CLASS CONTAINED;
    TYPE FLOAT
    {
        DEFAULT_VALUE -3.4E+38;
    }
    HANDLING READ & WRITE;
}

COMMAND Read__AnalogInputFB__LO_LIM
{
    BLOCK AnalogInputFB;
    INDEX 25;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            AnalogInputFB__LO_LIM
        }
    }
}

COMMAND Write__AnalogInputFB__LO_LIM
{
    BLOCK AnalogInputFB;
    INDEX 25;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            AnalogInputFB__LO_LIM
        }
        REPLY
        {
        }
    }
}

/* -----
* Temperature Technology Block
* -----*/

BLOCK TemperatureTB
{
    TYPE TRANSDUCER;
    NUMBER 1;
}

//----TTB-8----

```

```
VARIABLE TemperatureTB__PRIMARY_VALUE__Value
```

```
{
    LABEL [PRIMARY_VALUE__Value];
    HELP [TemperatureTB__PRIMARY_VALUE__Help];
    CLASS CONTAINED;
    TYPE FLOAT;
    HANDLING READ;
}
```

```
VARIABLE TemperatureTB__PRIMARY_VALUE__Status
```

```
{
    LABEL [PRIMARY_VALUE__Status];
    HELP [TemperatureTB__PRIMARY_VALUE__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (1)
    {
        DEF_STATUS
    }
    HANDLING READ;
}
```

```
COMMAND Read__TemperatureTB__PRIMARY_VALUE
```

```
{
    BLOCK TemperatureTB;
    INDEX 8;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            TemperatureTB__PRIMARY_VALUE__Value,
            TemperatureTB__PRIMARY_VALUE__Status
        }
    }
}
```

```
//----TTB-9----
```

```
VARIABLE TemperatureTB__PRIMARY_VALUE__UNIT
```

```
{
    LABEL [PRIMARY_VALUE__UNIT];
    HELP [TemperatureTB__PRIMARY_VALUE__UNIT__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (2)
    {
        DEFAULT_VALUE 1001;
        { 1000 ,[Kelvin], [Kelvin_help] },
        { 1001 ,[degree_Celsius], [degree_Celsius_help] },
        { 1002 ,[degree_Fahrenheit], [degree_Fahrenheit_help] },
        { 1003 ,[degree_Rankine], [degree_Rankine_help] },
        { 1243 ,[millivolt], [millivolt_help] },
        { 1281 ,[Ohm], [Ohm_help] }
    }
    HANDLING READ & WRITE;
}
```

```
COMMAND Read__TemperatureTB__PRIMARY_VALUE__UNIT
```

```
{
    BLOCK TemperatureTB;
```

```

INDEX 9;
OPERATION READ;
TRANSACTION
{
    REQUEST
    {
    }
    REPLY
    {
        TemperatureTB__PRIMARY_VALUE_UNIT
    }
}

COMMAND Write__TemperatureTB__PRIMARY_VALUE_UNIT
{
    BLOCK TemperatureTB;
    INDEX 9;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            TemperatureTB__PRIMARY_VALUE_UNIT
        }
        REPLY
        {
        }
    }
}

//-----TTB-10-----

VARIABLE TemperatureTB__SECONDARY_VALUE_1__Value
{
    LABEL [SECONDARY_VALUE_1__Value];
    HELP [TemperatureTB__SECONDARY_VALUE_1__Help];
    CLASS CONTAINED;
    TYPE FLOAT;
    HANDLING READ;
}

VARIABLE TemperatureTB__SECONDARY_VALUE_1__Status
{
    LABEL [SECONDARY_VALUE_1__Status];
    HELP [TemperatureTB__SECONDARY_VALUE_1__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (1)
    {
        DEF_STATUS
    }
    HANDLING READ;
}

COMMAND Read__TemperatureTB__SECONDARY_VALUE_1
{
    BLOCK TemperatureTB;
    INDEX 10;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {

```

```

    }
    REPLY
    {
        TemperatureTB__SECONDARY_VALUE_1__Value,
        TemperatureTB__SECONDARY_VALUE_1__Status
    }
}

//----TTB-11----

VARIABLE TemperatureTB__SECONDARY_VALUE_2__Value
{
    LABEL [SECONDARY_VALUE_2__Value];
    HELP [TemperatureTB__SECONDARY_VALUE_2__Help];
    CLASS CONTAINED;
    TYPE FLOAT;
    HANDLING READ;
}

VARIABLE TemperatureTB__SECONDARY_VALUE_2__Status
{
    LABEL [SECONDARY_VALUE_2__Status];
    HELP [TemperatureTB__SECONDARY_VALUE_2__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (1)
    {
        DEF_STATUS
    }
    HANDLING READ;
}

COMMAND Read__TemperatureTB__SECONDARY_VALUE_2
{
    BLOCK TemperatureTB;
    INDEX 11;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            TemperatureTB__SECONDARY_VALUE_2__Value,
            TemperatureTB__SECONDARY_VALUE_2__Status
        }
    }
}

//----TTB-14----

VARIABLE TemperatureTB__LIN_TYPE
{
    LABEL [LIN_TYPE];
    HELP [TemperatureTB__LIN_TYPE__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (1)
    {
        { 0 , [No_linearisation]],
        { 102 , "RTD Pt100 a=0.003850 (IEC 751, DIN 43760, JIS C1604-97,
BS1904)"},

```

```

        { 105 , "RTD Pt1000 a=0.003850 (IEC 751, DIN 43760, JIS C1604-
97, BS1904)"},
        { 123 , "RTD Ni100 a=0.006180 (DIN 43760)"},
        { 128 , "TC Type B, Pt30Rh-Pt6Rh (IEC 584, NIST MN 175, DIN
43710, BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 129 , "TC Type C (W5), W5-W26Rh (ASTM E 988)"},
        { 130 , "TC Type D (W3), W3-W25Rh (ASTM E 988)"},
        { 131 , "TC Type E, Ni10Cr-Cu45Ni (IEC584, NIST MN 175, DIN
43710, BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 133 , "TC Type J, Fe-Cu45Ni (IEC 584, NIST MN 175, DIN 43710,
BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 134 , "TC Type K, Ni10Cr-Ni5 (IEC 584, NIST MN 175, DIN 43710,
BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 135 , "TC Type N, Ni14CrSi-NiSi (IEC 584, NIST MN 175,
DIN 43710, BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 136 , "TC Type R, Pt13Rh-Pt (IEC 584, NIST MN 175, DIN 43710,
BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 137 , "TC Type S, Pt10Rh-Pt (IEC 584, NIST MN 175, DIN 43710,
BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 138 , "TC Type T, Cu-Cu45Ni (IEC 584, NIST MN 175, DIN 43710,
BS 4937, ANSI MC96.1, JIS C1602, NF C42-321)"},
        { 139 , "TC Type L, Fe-CuNi (DIN 43710)"},
        { 140 , "TC Type U, Cu-CuNi (DIN 43710)"},
        { 253 , [Ptxxx]}
    }
    HANDLING READ & WRITE;
}

```

```

COMMAND Read__TemperatureTB__LIN_TYPE
{
    BLOCK TemperatureTB;
    INDEX 14;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            TemperatureTB__LIN_TYPE
        }
    }
}

```

```

COMMAND Write__TemperatureTB__LIN_TYPE
{
    BLOCK TemperatureTB;
    INDEX 14;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            TemperatureTB__LIN_TYPE
        }
        REPLY
        {
        }
    }
}

```



```
//----TTB-36----
```

```
VARIABLE TemperatureTB__SENSOR_CONNECTION
```

```
{
    LABEL [SENSOR_CONNECTION];
    HELP [TemperatureTB__SENSOR_CONNECTION__Help];
    CLASS CONTAINED;
    TYPE ENUMERATED (1)
    {
        { 0 ,"2 wires" },
        { 1 ,"3 wires" },
        { 2 ,"4 wires" }

    }
    HANDLING READ & WRITE;
}
```

```
COMMAND Read__TemperatureTB__SENSOR_CONNECTION
```

```
{
    BLOCK TemperatureTB;
    INDEX 36;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            TemperatureTB__SENSOR_CONNECTION
        }
    }
}
```

```
COMMAND Write__TemperatureTB__SENSOR_CONNECTION
```

```
{
    BLOCK TemperatureTB;
    INDEX 36;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            TemperatureTB__SENSOR_CONNECTION
        }
        REPLY
        {
        }
    }
}
```

```
/* -----
*  MENU
*  -----*/
```

```
MENU Table_Main_Specialist
```

```
{
    LABEL [Main_MENU];
    ITEMS
    {
        Tab_s_Operator,
        Tab_s_Specialist
    }
}
```

```

}

MENU Tab_s_Operator
{
    LABEL [Menu_Operator];
    ITEMS
    {
        DeviceBlock __DEVICE_ID,
        AnalogInputFB __OUT__Value,
        AnalogInputFB __OUT__Status,
        AnalogInputFB __OUT__SCALE__UnitsIndex
    }
}

MENU Tab_s_Specialist
{
    LABEL [Menu_Specialist];
    ITEMS
    {
        Tab_s_AnalogInputFunctionBlock,
        Tab_s_TemperatureTechnologyBlock,
        Tab_s_DeviceBlock
    }
}

MENU Tab_s_AnalogInputFunctionBlock
{
    LABEL [Menu_AnalogInputFunctionBlock];
    ITEMS
    {
        AnalogInputFB __OUT__Value,
        AnalogInputFB __OUT__Status,
        AnalogInputFB __OUT__SCALE__EUat100percent,
        AnalogInputFB __OUT__SCALE__EUat0percent,
        AnalogInputFB __OUT__SCALE__UnitsIndex,
        AnalogInputFB __OUT__SCALE__DecimalPoint,
        AnalogInputFB __HI__LIM,
        AnalogInputFB __LO__LIM
    }
}

MENU Tab_s_TemperatureTechnologyBlock
{
    LABEL [Menu_TemperatureTechnologyBlock];
    ITEMS
    {
        TemperatureTB __PRIMARY_VALUE__Value,
        TemperatureTB __PRIMARY_VALUE__Status,
        TemperatureTB __PRIMARY_VALUE__UNIT,
        TemperatureTB __SECONDARY_VALUE_1__Value,
        TemperatureTB __SECONDARY_VALUE_1__Status,
        TemperatureTB __SECONDARY_VALUE_2__Value,
        TemperatureTB __SECONDARY_VALUE_2__Status,
        TemperatureTB __LIN_TYPE,
        TemperatureTB __SENSOR_CONNECTION
    }
}

MENU Tab_s_DeviceBlock
{
    LABEL [Menu_DeviceBlock];
    ITEMS
    {

```

```
DeviceBlock__DEVICE_MAN_ID,  
DeviceBlock__DEVICE_ID,  
DeviceBlock__DEVICE_SER_Num  
}  
}
```

IECNORM.COM: Click to view the full PDF of IEC 61804-3:2015

Annex D (normative)

Profiles of EDDL and Builtins

D.1 Conventions for profiles of EDDL and Builtins

Annex D provides the profiles of EDDL and Builtins. Profiles enable to select from this document the appropriate specifications for a specific profile. Clause D.1 provides conventions and contains templates for selection tables. The selection tables define which EDDL constructs and syntax are used and which Builtins are selected by the various consortia.

The following conventions in Table D.1, Table D.2, and Table D.3 for the profiles apply.

Table D.1 – Profile selection tables

Lexical construct	Attribute	Presence	Constraints

Table D.2 – EDDL Formal Definition profile tables

Reference	Title	Presence	Constraints

Table D.3 – Contents of selection tables

Column	Text	Meaning
Lexical construct		Is a lexical construct element of the EDDL.
Attributes	<text>	Is one of the attributes of a lexical construct element.
Presence	NO	This (sub)clause is not included in the profile.
	YES	This (sub)clause is fully (100%) included in the profile; in this case no further detail is given.
Constraints	—	No constraints other than those given in the reference document (sub)clause, or not applicable.
	<text>	The text defines the constraint directly; for longer text table footnotes or table notes may be used.
Reference	<#>	Clause or annex number.
Title	<text>	Title of heading.

For an EDD application it is not required to support all profiles, but a supported profile shall be supported completely.

D.2 Profiles for PROFIBUS and PROFINET

D.2.1 EDDL profile

Table D.4 is the selection of elements from the lexical structure constructs specified in Clause 7 and used by the consortium PROFIBUS&PROFINET International (PI). PROFIBUS and PROFINET⁷ are specified in IEC 61784-1 CPF3 and IEC 61784-2 CPF3.

Table D.4 – EDDL element selection for PROFIBUS and PROFINET

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.1	DD_REVISION		YES	—
7.2.2.2	DEVICE_REVISION		YES	—
7.2.2.3	DEVICE_TYPE		YES	—
7.2.2.4	EDD_PROFILE		YES	—
7.2.2.5	EDD_VERSION		YES	—
7.2.2.6	MANUFACTURER		YES	—
7.2.2.7	MANUFACTURER_EXT		YES	—
7.3	AXIS		YES	—
7.3.2.1		CONSTANT_UNIT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.3.2.2		MAX_VALUE	YES	—
7.3.2.3		MIN_VALUE	YES	—
7.3.2.4		SCALING	YES	—
7.43	BLOB		NO	—
7.4.1	BLOCK_A		NO	—
7.4.2	BLOCK_B		YES	—
7.4.2.2.1		NUMBER	YES	—
7.4.2.2.2		TYPE	YES	—
7.5	CHART		YES	—
7.36.10		MEMBERS	YES	—
7.5.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.5.2.2		LENGTH	YES	—
7.5.2.3		TYPE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.6	COLLECTION		YES	—

⁷ PROFIBUS and PROFINET are trade names of PROFIBUS Nutzerorganisation e.V. (PNO) as part of PROFIBUS and PROFINET International. PNO is a non-profit trade organization to support the fieldbus technologies PROFIBUS and PROFINET. This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the tradename. Use of the trade names requires permission from the trade name holder.

Reference	Basic construct	Attribute	Presence	Constraints
7.6.2		item-type	YES	Only item types supported by this communication profile.
7.36.10		MEMBERS	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.7	COMMAND		YES	—
7.7.2.1		OPERATION	YES	—
7.7.2.2		TRANSACTION	YES	—
7.7.2.3		INDEX	YES	—
7.7.2.4		BLOCK_B	YES	—
7.7.2.5		NUMBER	NO	—
7.7.2.6		SLOT	YES	—
7.7.2.7		SUB_SLOT	YES	PROFINET devices only.
7.7.2.11		API	YES	PROFINET devices only.
7.7.2.9		HEADER	NO	—
7.7.2.12		POST_REQUESTS RECEIVED	NO	—
7.36.14		RESPONSE_CODES	YES	—
7.7.2.12	COMPONENT		YES	—
7.36.9		LABEL	YES	—
7.8.2.11		BYTE_ORDER	YES	—
7.8.2.1		CAN_DELETE	YES	—
7.8.2.2		CHECK_CONFIGURATION	YES	—
7.36.1		CLASSIFICATION	YES	—
7.36.2		COMPONENT_PARENT	YES	—
7.36.3		COMPONENT_PATH	YES	—
7.8.2.3		COMPONENT_RELATIONS	YES	—
7.8.2.12		CONNECTION_POINT	YES	—
7.8.2.4		DECLARATION	NO	—
7.8.2.5		DETECT	YES	—
7.8.2.6		EDD	YES	—
7.36.8		HELP	YES	—
7.8.2.7		INITIAL_VALUES	YES	—
7.8.2.13		PRODUCT_URI	YES	—
7.36.13		PROTOCOL	YES	—
7.8.2.8		REDUNDANCY	YES	—
7.8.2.9		SCAN	YES	—
7.8.2.10		SCAN_LIST	YES	—
7.36.15		SUPPLIED_INTERFACE	NO	—
7.8.2.11	COMPONENT_FOLDER		YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.9		LABEL	YES	—
7.36.1		CLASSIFICATION	YES	—
7.36.2		COMPONENT_PARENT	YES	—
7.36.3		COMPONENT_PATH	YES	—
7.36.8		HELP	YES	—
7.36.13		PROTOCOL	YES	—
7.10	COMPONENT_REFERENCE		YES	—
7.36.1		CLASSIFICATION	YES	—
7.36.2		COMPONENT_PARENT	YES	—
7.36.3		COMPONENT_PATH	YES	—
7.2.2.2		DEVICE_REVISION	YES	—
7.2.2.3		DEVICE_TYPE	YES	—
7.2.2.6		MANUFACTURER	YES	—
7.36.13		PROTOCOL	YES	—
7.11	COMPONENT_RELATION		YES	—
7.11.2.1		COMPONENTS	YES	—
7.36.9		LABEL	YES	—
7.36.8		HELP	YES	—
7.11.2.2		RELATION_TYPE	YES	—
7.11.2.3		ADDRESSING	YES	—
7.11.2.4		MAXIMUM_NUMBER	YES	—
7.11.2.5		MINIMUM_NUMBER	YES	—
7.11.2.6		REQUIRED_INTERFACE	NO	—
7.36.15		SUPPLIED_INTERFACE	NO	—
7.14	EDIT_DISPLAY		NO	—
7.15	FILE		YES	—
7.36.12		MEMBERS	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.6		HANDLING	YES	—
7.36.21		IDENTITY	YES	—
7.15.2.1		SHARED	YES	—
7.15.2.2		ON_UPDATE_ACTIONS	NO	—
7.16	GRAPH		YES	—
7.36.12		MEMBERS	YES	—
7.16.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.16.2.2		X_AXIS	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.17	GRID		YES	—
7.17.2.1		VECTORS	YES	—
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.17.2.2		ORIENTATION	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.18	IMAGE		YES	—
7.18.2.1		PATH	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.18.2.2		LINK	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.19	IMPORT		YES	—
7.20	INTERFACE		NO	—
7.21	LIKE		YES	—
7.22	LIST		YES	—
7.22.2.1		TYPE	YES	—
7.22.2.2		CAPACITY	YES	—
7.22.2.3		COUNT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.23	MENU		YES	—
7.23.2.1		ITEMS	YES	Except: BLOCK_A parameters, elements of BLOCK_A parameters, qualifier DISPLAY_VALUE
7.36.9		LABEL	YES	—
7.23.2.2		ACCESS	YES	—
7.23.2.12		EXIT_ACTIONS	YES	—
7.36.8		HELP	YES	—
7.23.2.13		INIT_ACTIONS	YES	—
7.23.2.3		POST_EDIT_ACTIONS	YES	—
7.23.2.4		POST_READ_ACTIONS	YES	—
7.23.2.5		POST_WRITE_ACTIONS	YES	—
7.23.2.6		PRE_EDIT_ACTIONS	YES	—
7.23.2.7		PRE_READ_ACTIONS	YES	—
7.23.2.8		PRE_WRITE_ACTIONS	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.23.2.11		STYLE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.24	METHOD		YES	—
7.24.1		method_parameter	YES	—
7.36.1		DEFINITION	YES	—
7.24.2.1		ACCESS	YES	—
7.24.2.2		CLASS	YES	Except: HART
7.36.9		LABEL	YES	—
7.36.8		HELP	YES	—
7.36.18		PRIVATE	YES	—
7.24.2.3		TYPE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.42	PLUGIN		YES	—
7.36.9		LABEL	YES	—
7.36.8		HELP	YES	—
7.42.2		UUID	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.26	RECORD		NO	—
7.27	REFERENCE_ARRAY		YES	—
7.6.2		item_type	YES	Only item types supported by this communication profile.
7.27.2		ELEMENTS	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.28.1	REFRESH		YES	—
7.28.2	UNIT		YES	—
7.28.3	WRITE_AS_ONE		YES	—
7.36.14	RESPONSE_CODES		YES	—
7.30	SOURCE		YES	—
7.36.12		MEMBERS	YES	—
7.36.5		EMPHASIS	YES	—
7.30.2.1		EXIT_ACTIONS	YES	—
7.36.8		HELP	YES	—
7.30.2.2		INIT_ACTIONS	YES	—
7.36.9		LABEL	YES	—
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—
7.30.2.3		REFRESH_ACTIONS	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.30.2.4		Y_AXIS	YES	—
7.31	TEMPLATE		YES	—
7.31.2		DEFAULT_VALUES	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.16		VALIDITY	YES	—
7.32	VALUE_ARRAY		YES	—
7.36.9		LABEL	YES	—
7.32.2.1		NUMBER_OF_ELEMENTS	YES	—
7.32.2.2		TYPE	YES	—
7.36.8		HELP	YES	—
7.36.14		RESPONSE_CODES	NO	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.20		WRITE_MODE	NO	—
7.33	VARIABLE		YES	—
7.33.2.1		CLASS	YES	Except: HART
7.36.9		LABEL	YES	—
7.33.2.2		TYPE	YES	—
7.33.2.3		CONSTANT_UNIT	YES	—
7.33.2.4		DEFAULT_VALUE	YES	—
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.33.2.5		INITIAL_VALUE	YES	—
7.33.2.6		POST_EDIT_ACTIONS	YES	—
7.33.2.7		POST_READ_ACTIONS	YES	—
7.33.2.15		POST_RQSTUPDATE_ACTIONS	NO	—
7.33.2.16		POST_USERCHANGE_ACTIONS	NO	—
7.33.2.8		POST_WRITE_ACTIONS	YES	—
7.33.2.9		PRE_EDIT_ACTIONS	YES	—
7.33.2.10		PRE_READ_ACTIONS	YES	—
7.33.2.11		PRE_WRITE_ACTIONS	YES	—
7.36.18		PRIVATE	YES	—
7.33.2.13		REFRESH_ACTIONS	YES	—
7.36.14		RESPONSE_CODES	NO	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.20		WRITE_MODE	NO	—
7.33.2.15	VARIABLE_LIST		NO	—
7.35	WAVEFORM		YES	—
7.35.2.1		TYPE	YES	—
7.36.5		EMPHASIS	YES	—
7.35.2.2		EXIT_ACTIONS	YES	—
7.36.6		HANDLING	YES	—
7.36.8		HELP	YES	—
7.35.2.3		INIT_ACTIONS	YES	—
7.35.2.4		KEY_POINTS	YES	—
7.36.9		LABEL	YES	—
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—
7.35.2.5		REFRESH_ACTIONS	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.35.2.6		Y_AXIS	YES	—
7.36.21	Conditional expressions		YES	—
7.38	Referencing		YES	—
7.39	Strings		YES	—
7.40	Expressions		YES	—
7.41	Text dictionary		YES	—
7.33.2.2.2.2	DATE_AND_TIME		YES	—
7.33.2.2.2.8	BOOLEAN		YES	Coding: TRUE is represented by the value 0xFF
NOTE Text is only ISO Latin-1.				

D.2.2 Builtin profile

The Builtin profiles are specified in IEC 61804-5.

D.2.3 EDDL Formal Definition profile

Reference	Title	Presence	Constraints
A.1	EDDL preprocessor	YES	—
A.2	Conventions	YES	—
A.3	Operators	YES	—
A.4	Keywords	YES	—
A.5	Terminals	YES	—
A.6	Formal EDDL syntax	YES	—

D.3 Profiles for FOUNDATION™ fieldbus

D.3.1 EDDL profile

Table D.5 is the selection of elements from the lexical structure constructs specified in Clause 7 and used by the consortium Fieldbus Foundation.

Table D.5 – EDDL element selection for FOUNDATION fieldbus

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.1		DD_REVISION	YES	—
7.2.2.2		DEVICE_REVISION	YES	—
7.2.2.3		DEVICE_TYPE	YES	May not be an identifier; shall be an integer.
7.2.2.4		EDD_PROFILE	NO	—
7.2.2.5		EDD_VERSION	NO	—
7.2.2.6		MANUFACTURER	YES	May not be an identifier; shall be an integer.
7.2.2.7		MANUFACTURER_EXT	NO	—
7.3	AXIS		YES	—
7.3.2.1		CONSTANT_UNIT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.3.2.2		MAX_VALUE	YES	—
7.3.2.3		MIN_VALUE	YES	—
7.3.2.4		SCALING	YES	—
7.43	BLOB		NO	—
7.4.1	BLOCK_A		YES	—
7.4.1.2.1		CHARACTERISTICS	YES	—
7.36.9		LABEL	YES	Only string_constant.
7.4.1.2.2		PARAMETERS	YES	—
7.4.1.2.3		AXIS_ITEMS	YES	—
7.4.1.2.4		CHART_ITEMS	YES	—
7.4.1.2.5		COLLECTION_ITEMS	YES	—
7.4.1.2.6		EDIT_DISPLAY_ITEMS	YES	—
7.4.1.2.7		FILE_ITEMS	YES	—
7.4.1.2.8		GRAPH_ITEMS	YES	—
7.4.1.2.9		GRID_ITEMS	YES	—
7.4.1.2.10		IMAGE_ITEMS	YES	—
7.4.1.2.11		LIST_ITEMS	YES	—
7.4.1.2.12		LOCAL_PARAMETERS	YES	—
		HELP	YES	Only string_constant
7.4.1.2.13		MENU_ITEMS	YES	—
7.4.1.2.14		METHOD_ITEMS	YES	—
7.4.1.2.29		PLUGIN_ITEMS	YES	—
7.4.1.2.15		PARAMETER_LISTS	YES	—
7.4.1.2.16		REFERENCE_ARRAY_ITEMS	YES	Referred as ITEM_ARRAY_ITEMS.

Reference	Basic construct	Attribute	Presence	Constraints
7.4.1.2.17		REFRESH_ITEMS	YES	—
7.4.1.2.18		SOURCE_ITEMS	YES	—
7.4.1.2.19		UNIT_ITEMS	YES	—
7.4.1.2.20		WAVEFORM_ITEMS	YES	—
7.4.1.2.21		WRITE_AS_ONE_ITEMS	YES	—
7.4.1.2.22		CHARTS	YES	—
7.4.1.2.28		FILES	YES	—
7.4.1.2.23		LISTS	YES	—
7.4.1.2.24		GRAPHS	YES	—
7.4.1.2.25		GRIDS	YES	—
7.4.1.2.26		MENUS	YES	—
7.4.1.2.27		METHODS	YES	—
7.4.1.2.30		PLUGINS	YES	—
7.4.2	BLOCK_B		NO	—
7.5	CHART		YES	—
7.36.10		MEMBERS	YES	—
7.5.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.5.2.2		LENGTH	YES	—
7.5.2.3		TYPE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.6	COLLECTION		YES	—
7.6.2		item-type	YES	Only item types supported by this communication profile.
7.36.10		MEMBERS	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	NO	—
7.36.19		VISIBILITY	YES	—
7.7	COMMAND		NO	—
7.7.2.12	COMPONENT		NO	—
7.8.2.11	COMPONENT_FOLDER		NO	—
7.10	COMPONENT_REFERENCE		NO	—
7.11	COMPONENT_RELATION		NO	—
7.14	EDIT_DISPLAY		YES	—
7.14.2.1		EDIT_ITEMS	YES	—
7.36.9		LABEL	YES	Only string_constant
7.14.2.2		DISPLAY_ITEMS	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.14.2.3		POST_EDIT_ACTIONS	YES	Except: DEFINITION
7.14.2.4		PRE_EDIT_ACTIONS	YES	Except: DEFINITION
7.15	FILE		YES	—
7.36.12		MEMBERS	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.36.6		HANDLING	NO	—
7.36.21		IDENTITY	NO	—
7.15.2.1		SHARED	NO	—
7.15.2.2		ON_UPDATE_ACTIONS	NO	—
7.16	GRAPH		YES	—
7.36.12		MEMBERS	YES	—
7.16.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.16.2.2		X_AXIS	YES	—
7.17	GRID		YES	—
7.17.2.1		VECTORS	YES	—
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.17.2.2		ORIENTATION	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.18	IMAGE		YES	—
7.18.2.1		PATH	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.18.2.2		LINK	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.19	IMPORT		YES	—
7.20	INTERFACE		NO	—
7.21	LIKE		YES	The optional item-type shall be used.
7.22	LIST		YES	—
7.22.2.1		TYPE	YES	—
7.22.2.2		CAPACITY	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.22.2.3		COUNT	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.23	MENU		YES	—
7.23.2.1		ITEMS	YES	—
7.36.9		LABEL	YES	Only string_constant
7.23.2.2		ACCESS	NO	—
7.23.2.12		EXIT_ACTIONS	YES	—
7.36.8		HELP	YES	Only string_constant
7.23.2.13		INIT_ACTIONS	YES	—
7.23.2.3		POST_EDIT_ACTIONS	NO	Except: DEFINITION
7.23.2.4		POST_READ_ACTIONS	YES	Except: DEFINITION; only on upload from device menus.
7.23.2.5		POST_WRITE_ACTIONS	YES	Except: DEFINITION; only on download from device menus.
7.23.2.6		PRE_EDIT_ACTIONS	NO	Except: DEFINITION
7.23.2.7		PRE_READ_ACTIONS	YES	Except: DEFINITION; only on upload from device menus.
7.23.2.8		PRE_WRITE_ACTIONS	YES	Except: DEFINITION; only on download from device menus.
7.23.2.11		STYLE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.24	METHOD		YES	—
7.24.1		method_parameter	YES	—
7.36.4		DEFINITION	YES	—
7.24.2.1		ACCESS	NO	—
7.24.2.2		CLASS	YES	—
7.36.9		LABEL	YES	Only string_constant
7.36.8		HELP	YES	Only string_constant
7.36.18		PRIVATE	YES	—
7.24.2.3		TYPE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.42	PLUGIN		YES	—
7.36.9		LABEL	YES	Only string_constant
7.36.8		HELP	YES	Only string_constant
7.42.2		UUID	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.26	RECORD		YES	—
7.36.10		MEMBERS	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.9		LABEL	YES	Only string_constant
7.36.8		HELP	YES	Only string_constant
7.36.18		PRIVATE	YES	—
7.36.14		RESPONSE_CODES	YES	Only: referenced RESPONSE_CODES
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.20		WRITE_MODE	YES	—
7.27	REFERENCE_ARRAY		YES	Referred as ITEM_ARRAY
7.6.2		item-type	YES	Only item types supported by this communication profile.
7.27.2		ELEMENTS	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.28.1	REFRESH		YES	—
7.28.2	UNIT		YES	—
7.28.3	WRITE_AS_ONE		YES	—
7.36.14	RESPONSE_CODES		YES	—
7.30	SOURCE		YES	—
7.36.12		MEMBERS	YES	—
7.36.5		EMPHASIS	YES	—
7.30.2.1		EXIT_ACTIONS	YES	—
7.36.8		HELP	YES	Only string_constant
7.30.2.2		INIT_ACTIONS	YES	—
7.36.9		LABEL	YES	Only string_constant
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—
7.30.2.3		REFRESH_ACTIONS	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.30.2.4		Y_AXIS	YES	—
7.31	TEMPLATE		YES	—
7.31.2		DEFAULT_VALUES	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.36.16		VALIDITY	YES	—
7.30.2.4	VALUE_ARRAY		YES	—
7.36.9		LABEL	YES	Only string_constant
7.32.2.1		NUMBER_OF_ELEMENTS	YES	Shall evaluate to a constant value.
7.32.2.2		TYPE	YES	—
7.36.8		HELP	YES	Only string_constant

Reference	Basic construct	Attribute	Presence	Constraints
7.36.18		PRIVATE	YES	—
7.36.14		RESPONSE_CODES	YES	Only: referenced RESPONSE_CODES
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.20		WRITE_MODE	YES	—
7.33	VARIABLE		YES	—
7.33.2.1		CLASS	YES	Every VARIABLE shall have one and only one of the following classes: CONTAINED, INPUT, OUTPUT Additional classes can be combined: ALARM, DIAGNOSTIC, DYNAMIC, LOCAL, OPERATE, SERVICE, SPECIALIST, TUNE,
7.36.9		LABEL	YES	Mandatory attribute. Only string_constant
7.33.2.2		TYPE	YES	Except: DATE, OBJECT REFERENCE, PACKED_ASCII. Only MIN_VALUE, MAX_VALUE options for DATE_AND_TIME, TIME, DURATION and TIME_VALUE datatypes. For arithmetic and string types enumerations are not supported. For TIME_VALUE the default size is 8 octets. The 4-octet TIME_VALUE is not supported. DEFAULT_VALUE and INITIAL_VALUE are not declared inside the TYPE specifier.
7.33.2.3		CONSTANT_UNIT	YES	—
7.33.2.4		DEFAULT_VALUE	YES	—
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant
7.33.2.5		INITIAL_VALUE	NO	—
7.33.2.6		POST_EDIT_ACTIONS	YES	Except: DEFINITION
7.33.2.7		POST_READ_ACTIONS	YES	Except: DEFINITION
7.33.2.15		POST_RQSTUPDATE_ACTIONS	NO	—
7.33.2.16		POST_USERCHANGE_ACTIONS	NO	—
7.33.2.8		POST_WRITE_ACTIONS	YES	Except: DEFINITION
7.33.2.9		PRE_EDIT_ACTIONS	YES	Except: DEFINITION
7.33.2.10		PRE_READ_ACTIONS	YES	Except: DEFINITION
7.33.2.11		PRE_WRITE_ACTIONS	YES	Except: DEFINITION
7.36.18		PRIVATE	YES	—
7.33.2.13		REFRESH_ACTIONS	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.14		RESPONSE_CODES	YES	Only: referenced RESPONSE_CODES
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.36.20		WRITE_MODE	YES	—
7.33.2.15	VARIABLE_LIST		YES	—
7.36.10		MEMBERS	YES	—
7.36.8		HELP	YES	Only string_constant
7.36.9		LABEL	YES	Only string_constant
7.36.14		RESPONSE_CODES	YES	Only: referenced RESPONSE_CODES
7.35	WAVEFORM		YES	—
7.35.2.1		TYPE	YES	—
7.36.5		EMPHASIS	YES	—
7.35.2.2		EXIT_ACTIONS	YES	—
7.36.6		HANDLING	YES	—
7.36.8		HELP	YES	Only string_constant
7.35.2.3		INIT_ACTIONS	YES	—
7.35.2.4		KEY_POINTS	YES	—
7.36.9		LABEL	YES	Only string_constant
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—
7.35.2.5		REFRESH_ACTIONS	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.35.2.6		Y_AXIS	YES	—
7.36.21	Conditional expressions		YES	—
7.38	References		YES	The numbering of elements for LIST and VALUE_ARRAY starts with 1.
7.39	Strings		YES	—
7.40	Expressions		YES	Except: ARRAY_INDEX (see Table 250) and attribute values of VARIABLES: INITIAL_VALUE, VARIABLE_STATUS (see Table 251) are not possible.
7.41	Text dictionary		YES	—
7.33.2.2.2.2	DATE_AND_TIME		YES	Coding: the actual year is coded without century, i.e.0...99 years.
7.33.2.2.2.8	BOOLEAN		YES	Coding: TRUE is represented by the value 0xFF

D.3.2 Builtin profile

The Builtin profiles are specified in IEC 61804-5.

D.3.3 EDDL Formal Definition profile

Reference	Title	Presence	Constraints
A.1	EDDL preprocessor	YES	—
A.2	Conventions	YES	—
A.3	Operators	YES	—
A.4	Keywords	YES	—
A.5	Terminals	YES	—
A.6	Formal EDDL syntax	YES	—

D.4 Profiles for HART® Communication Foundation (HCF)**D.4.1 EDDL profile**

Table D.6 is the selection of elements from the lexical structure constructs specified in Clause 7 and used by the consortium HART® Communication Foundation.

Table D.6 – EDDL element selection for HCF

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.1	DD_REVISION		YES	One octet.
7.2.2.2	DEVICE_REVISION		YES	One octet.
7.2.2.3	DEVICE_TYPE		YES	Two octets.
7.2.2.4	EDD_PROFILE		NO	—
7.2.2.5	EDD_VERSION		NO	—
7.2.2.6	MANUFACTURER		YES	Two octets.
7.2.2.7	MANUFACTURER_EXT		NO	—
7.3	AXIS		YES	—
7.3.2.1		CONSTANT_UNIT	YES	Not conditional.
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.3.2.2		MAX_VALUE	YES	—
7.3.2.3		MIN_VALUE	YES	—
7.3.2.4		SCALING	YES	—
7.43	BLOB		YES	—
7.36.6		HANDLING	YES	—
7.36.8		HELP	YES	—
7.36.21		IDENTITY	YES	—
7.36.9		LABEL	YES	—
7.4	BLOCK		NO	—
7.5	CHART		YES	—
7.36.10		MEMBERS	YES	—
7.5.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.5.2.2		LENGTH	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.5.2.3		TYPE	YES	Only GAUGE, HORIZONTAL_BAR, SCOPE, STRIP, SWEEP, VERTICAL_BAR
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.6	COLLECTION		YES	
7.6.2		item-type	YES	Use is discouraged, supported for backward compatibility. Only item-types: VARIABLE, ARRAY, COLLECTION, LIST, MENU, EDIT_DISPLAY, GRAPH, CHART, GRID, IMAGE, METHOD, PLUGIN, FILE
7.36.10		MEMBERS	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.7	COMMAND		YES	—
7.7.2.1		OPERATION	YES	Only: READ, WRITE, COMMAND, no conditionals.
7.7.2.2		TRANSACTION	YES	No conditionals. REQUEST and REPLY may only contain VARIABLE references; LIST, ARRAY, COLLECTION not supported.
7.7.2.3		INDEX	NO	—
7.7.2.4		BLOCK_B	NO	—
7.7.2.5		NUMBER	YES	Only unsigned_integer.
7.7.2.6		SLOT	NO	—
7.7.2.7		SUB_SLOT	NO	—
7.7.2.11		API	NO	—
7.7.2.9		HEADER	NO	—
7.7.2.12		POST_RQSTRECEIVE_ACTIONS	YES	Except: DEFINITION
7.36.14		RESPONSE_CODES	YES	Only: embedded RESPONSE_CODES; no conditionals.
7.7.2.12	COMPONENT		NO	—
7.8.2.11	COMPONENT_FOLDER		NO	—
7.10	COMPONENT_REFERENCE		NO	—
7.11	COMPONENT_RELATION		NO	—
7.14	EDIT_DISPLAY		YES	—
7.14.2.1		EDIT_ITEMS	YES	Only: VARIABLE, WRITE_AS_ONE
7.36.9		LABEL	YES	—
7.14.2.2		DISPLAY_ITEMS	YES	Only: VARIABLE
7.14.2.3		POST_EDIT_ACTIONS	YES	Except: DEFINITION

Reference	Basic construct	Attribute	Presence	Constraints
7.14.2.4		PRE_EDIT_ACTIONS	YES	Except: DEFINITION
7.15	FILE		YES	—
7.36.12		MEMBERS	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.36.6		HANDLING	YES	—
7.36.21		IDENTITY	YES	—
7.15.2.1		SHARED	YES	—
7.15.2.2		ON_UPDATE_ACTIONS	YES	Except: DEFINITION
7.16	GRAPH		YES	—
7.36.12		MEMBERS	YES	—
7.16.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.16.2.2		X_AXIS	YES	—
7.17	GRID		YES	—
7.17.2.1		VECTORS	YES	—
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant,
7.36.9		LABEL	YES	Only string_constant.
7.17.2.2		ORIENTATION	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.18	IMAGE		YES	—
7.18.2.1		PATH	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.18.2.2		LINK	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.19	IMPORT		YES	EDD cannot be referenced by file name.
7.20	INTERFACE		NO	—
7.21	LIKE		YES	—
7.22	LIST		YES	—
7.22.2.1		TYPE	YES	—
7.22.2.2		CAPACITY	YES	—
7.22.2.3		COUNT	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	NO	—
7.23	MENU		YES	—
7.23.2.1		ITEMS	YES	Only: MENU, METHOD, VARIABLE, string_constant, EDIT_DISPLAY, , COLLECTION, GRAPH, CHART, IMAGE, GRID, ROWBREAK, COLUMNBREAK, and PLUGIN For variable-reference qualifiers DISPLAY_VALUE, READ_ONLY, NO_LABEL, and NO_UNIT are supported. For menu-reference REVIEW is supported. For image-reference INLINE, ALIGN_LEFT, ALIGN_RIGHT and NO_LABEL are supported. For method-references, no parameters are allowed. Only the method LABEL is displayed.
7.36.9		LABEL	YES	Only string_constant.
7.23.2.2		ACCESS	NO	—
7.23.2.12		EXIT_ACTIONS	NO	—
7.36.8		HELP	YES	Only string_constant.
7.23.2.13		INIT_ACTIONS	NO	—
7.23.2.3		POST_EDIT_ACTIONS	NO	—
7.23.2.4		POST_READ_ACTIONS	YES	Only on upload from device menus.
7.23.2.5		POST_WRITE_ACTIONS	YES	Only on download to device menus.
7.23.2.6		PRE_EDIT_ACTIONS	NO	—
7.23.2.7		PRE_READ_ACTIONS	YES	Only on upload from device menus.
7.23.2.8		PRE_WRITE_ACTIONS	YES	Only on download to device menus.
7.23.2.11		STYLE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.24	METHOD		YES	—
7.24.1		method_parameter	YES	—
7.36.1		DEFINITION	YES	DD_STRING supports the following operators: + =
7.24.2.1		ACCESS	NO	—
7.24.2.2		CLASS	YES	Only SPECIALIST Supported for backward compatibility: DIAGNOSTIC, SERVICE, ANALOG_OUTPUT, COMPUTATION, CORRECTION, DEVICE, DISCRETE, FREQUENCY, HART, LOCAL_DISPLAY, and INPUT
7.36.9		LABEL	YES	Only string_constant.
7.36.8		HELP	YES	Only string_constant.

Reference	Basic construct	Attribute	Presence	Constraints
7.36.18		PRIVATE	YES	—
7.24.2.3		TYPE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.42	PLUGIN		YES	—
7.36.9		LABEL	YES	Only string_constant.
7.36.8		HELP	YES	Only string_constant.
7.42.2		UUID	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.26	RECORD		NO	—
7.27	REFERENCE_ARRAY		YES	Concrete syntax: ARRAY
7.6.2		item-type	YES	Only: VARIABLE, ARRAY, COLLECTION, LIST, MENU, EDIT_DISPLAY, GRAPH, CHART, COMMAND, METHOD, FILE
7.27.2		ELEMENTS	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	NO	—
7.28.1	REFRESH		YES	—
7.28.2	UNIT		YES	—
7.28.3	WRITE_AS_ONE		YES	—
7.36.14	RESPONSE_CODES		NO	—
7.30	SOURCE		YES	—
7.36.12		MEMBERS	YES	—
7.36.5		EMPHASIS	YES	—
7.30.2.1		EXIT_ACTIONS	YES	—
7.36.8		HELP	YES	Only string_constant.
7.30.2.2		INIT_ACTIONS	YES	—
7.36.9		LABEL	YES	Only string_constant.
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—
7.30.2.3		REFRESH_ACTIONS	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	NO	—
7.30.2.4		Y_AXIS	YES	—
7.31	TEMPLATE		YES	—
7.31.2		DEFAULT_VALUES	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.9		LABEL	YES	Only string_constant.
7.36.16		VALIDITY	YES	—
7.32	VALUE_ARRAY		YES	Concrete syntax: ARRAY

Reference	Basic construct	Attribute	Presence	Constraints
7.36.9		LABEL	YES	Only string_constant.
7.32.2.1		NUMBER_OF_ELEMENTS	YES	Only unsigned_integer.
7.32.2.2		TYPE	YES	—
7.36.8		HELP	YES	Only string_constant.
7.36.14		RESPONSE_CODES	NO	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	NO	—
7.36.20		WRITE_MODE	NO	—
7.33	VARIABLE		YES	—
7.33.2.1		CLASS	YES	Optional attribute. Only: ALARM, ANALOG_OUTPUT (ANALOG_CHANNEL), COMPUTATION, CORRECTION, DEVICE_DISCRETE, FREQUENCY, HART, LOCAL_DISPLAY, MODE, INPUT (RANGE), SPECIALIST, TUNE, If not CLASS LOCAL then can be combined with DYNAMIC, DIAGNOSTIC, SERVICE, and IS_CONFIG If CLASS LOCAL, then cannot be combined with any other CLASS
7.36.9		LABEL	YES	Only string_constant.
7.33.2.2		TYPE	YES	Only: FLOAT, DOUBLE, INTEGER, UNSIGNED_INTEGER, ENUMERATED, BIT_ENUMERATED, INDEX, PACKED_ASCII, ASCII, PASSWORD, DATE, TIME_VALUE For arithmetic and string types enumerations are not supported. No conditionals type. Initial value of all PACKED_ASCII, ASCII and PASSWORD is a string with each character set to '?'. For TIME_VALUE the default size is 4 octets. The 8-octet TIME_VALUE is not supported. DEFAULT_VALUE is not declared inside the TYPE specifier.
7.33.2.3		CONSTANT_UNIT	YES	No conditionals.
7.33.2.4		DEFAULT_VALUE	YES	Only constant-expression.
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	Only string_constant.
7.33.2.5		INITIAL_VALUE	NO	—
7.33.2.6		POST_EDIT_ACTIONS	YES	Except: DEFINITION
7.33.2.7		POST_READ_ACTIONS	YES	Except: DEFINITION
7.33.2.15		POST_READ_UPDATE_ACTIONS	YES	Except: DEFINITION

Reference	Basic construct	Attribute	Presence	Constraints
7.33.2.16		POST_USERCHANGE_ACTIONS	YES	Except: DEFINITION
7.33.2.8		POST_WRITE_ACTIONS	YES	Except: DEFINITION
7.33.2.9		PRE_EDIT_ACTIONS	YES	Except: DEFINITION
7.33.2.10		PRE_READ_ACTIONS	YES	Except: DEFINITION
7.33.2.11		PRE_WRITE_ACTIONS	YES	Except: DEFINITION
7.36.18		PRIVATE	YES	—
7.33.2.13		REFRESH_ACTIONS	YES	—
7.36.14		RESPONSE_CODES	NO	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.36.20		WRITE_MODE	NO	—
7.33.2.15	VARIABLE_LIST		NO	—
7.35	WAVEFORM		YES	—
7.35.2.1		TYPE	YES	—
7.36.5		EMPHASIS	YES	—
7.35.2.2		EXIT_ACTIONS	YES	—
7.36.6		HANDLING	YES	—
7.36.8		HELP	YES	Only string_constant.
7.35.2.3		INIT_ACTIONS	YES	—
7.35.2.4		KEY_POINTS	YES	—
7.36.9		LABEL	YES	Only string_constant.
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—
7.35.2.5		REFRESH_ACTIONS	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	NO	—
7.35.2.6		Y_AXIS	YES	—
7.36.21	Conditional expressions		YES	—
7.38	Referencing		YES	References to DD-Items, array and list elements, collection and file members, bits from bit-enumerated variables, plus only the following attribute references: DEFAULT_VALUE, IDENTITY, MAX_VALUE, MAX_VALUE1, MAX_VALUE2, MAX_VALUE3, MAX_VALUE4, MAX_VALUE5, MAX_VALUE6, MAX_VALUE7, MAX_VALUE8, MAX_VALUE9, MIN_VALUE, MIN_VALUE1, MIN_VALUE2, MIN_VALUE3, MIN_VALUE4, MIN_VALUE5, MIN_VALUE6, MIN_VALUE7, MIN_VALUE8, MIN_VALUE9, HELP, LABEL, plus COUNT, CAPACITY, FIRST and LAST with LIST, plus VIEW_MIN, VIEW_MAX with AXIS

Reference	Basic construct	Attribute	Presence	Constraints
7.39	Strings		YES	Only: string literals, string variable-references, dictionary references, enumeration variable value references. Assignments on strings cannot be done directly, only by using the string related builtins. String operators are not supported.
7.40	Expressions		YES	Only numeric expressions including: constant, parenthesized expression, variable reference, attribute values of VARIABLES (DEFAULT_VALUE, MAX_VALUE, MAX_VALUE1, MAX_VALUE2, MAX_VALUE3, MAX_VALUE4, MAX_VALUE5, MAX_VALUE6, MAX_VALUE7, MAX_VALUE8, MAX_VALUE9, MIN_VALUE, MIN_VALUE1, MIN_VALUE2, MIN_VALUE3, MIN_VALUE4, MIN_VALUE5, MIN_VALUE6, MIN_VALUE7, MIN_VALUE8, MIN_VALUE9), attribute values of an AXIS (VIEW_MIN, VIEW_MAX), attribute values of a LIST (FIRST, LAST, COUNT, CAPACITY)
7.41	Text dictionary		YES	Only: HART Standard Dictionary plus one per manufacturer.
7.33.2.2.2.2	DATE_AND_TIME		NO	—
7.33.2.2.2.8	BOOLEAN		NO	—
NOTE Text is only ISO Latin-1 or UTF-8. Katakana, ([k]) is entered as Romaji and is a series of English characters. Two to three characters per Katakana symbol.				

D.4.2 Builtin profile

The Builtin profiles are specified in IEC 61804-5.

D.4.3 EDDL Formal Definition profile

Reference	Title	Presence	Constraints
A.1	EDDL preprocessor	YES	—
A.2	Conventions	YES	—
A.3	Operators	YES	—
A.4	Keywords	YES	—
A.5	Terminals	YES	—
A.6	Formal EDDL syntax	YES	—

D.5 Profiles for Communication Servers

D.5.1 EDDL profile

Table D.7 is the selection of elements from the lexical structure constructs specified in Clause 7 and used for the integration of Communication Servers.

Table D.7 – EDDL element selection for Communication Servers

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.1	DD_REVISION		YES	—
7.2.2.2	DEVICE_REVISION		YES	—
7.2.2.3	DEVICE_TYPE		YES	—
7.2.2.4	EDD_PROFILE		YES	—
7.2.2.5	EDD_VERSION		YES	—
7.2.2.6	MANUFACTURER		YES	—
7.2.2.7	MANUFACTURER_EXT		YES	—
7.3	AXIS		YES	—
7.3.2.1		CONSTANT_UNIT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.3.2.2		MAX_VALUE	YES	—
7.3.2.3		MIN_VALUE	YES	—
7.3.2.4		SCALING	YES	—
7.43	BLOB		NO	—
7.4.1	BLOCK_A		NO	—
7.4.2	BLOCK_B		YES	—
7.4.2.2.1		NUMBER	YES	—
7.4.2.2.2		TYPE	YES	—
7.5	CHART		YES	—
7.36.10		MEMBERS	YES	—
7.5.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.5.2.2		LENGTH	YES	—
7.5.2.3		TYPE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.6	COLLECTION		YES	—
7.6.2		item-type	YES	Only item types supported by this communication profile.
7.36.10		MEMBERS	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.7	COMMAND		YES	—
7.7.2.1		OPERATION	YES	—
7.7.2.2		TRANSACTION	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.7.2.3		INDEX	YES	—
7.7.2.4		BLOCK_B	YES	—
7.7.2.5		NUMBER	NO	—
7.7.2.6		SLOT	YES	—
7.7.2.7		SUB_SLOT	YES	—
7.7.2.11		API	YES	—
7.7.2.9		HEADER	NO	—
7.7.2.12		POST_RQSTRECEIVE_ACTIONS	NO	—
7.36.14		RESPONSE_CODES	YES	—
7.7.2.12	COMPONENT		YES	—
7.36.9		LABEL	YES	—
7.8.2.11		BYTE_ORDER	YES	—
7.8.2.1		CAN_DELETE	YES	—
7.8.2.2		CHECK_CONFIGURATION	YES	—
7.36.1		CLASSIFICATION	YES	—
7.36.2		COMPONENT_PARENT	YES	—
7.36.3		COMPONENT_PATH	YES	—
7.8.2.3		COMPONENT_RELATIONS	YES	—
7.8.2.12		CONNECTION_POINT	YES	—
7.8.2.4		DECLARATION	NO	—
7.8.2.5		DETECT	YES	—
7.8.2.6		EDD	YES	—
7.36.8		HELP	YES	—
7.8.2.7		INITIAL_VALUES	YES	—
7.8.2.13		PRODUCT_URI	YES	—
7.36.13		PROTOCOL	YES	—
7.8.2.8		REDUNDANCY	YES	—
7.8.2.9		SCAN	YES	—
7.8.2.10		SCAN_LIST	YES	—
7.36.15		SUPPLIED_INTERFACE	NO	—
7.8.2.11	COMPONENT_FOLDER		YES	—
7.36.9		LABEL	YES	—
7.36.1		CLASSIFICATION	YES	—
7.36.2		COMPONENT_PARENT	YES	—
7.36.3		COMPONENT_PATH	YES	—
7.36.8		HELP	YES	—
7.36.13		PROTOCOL	YES	—
7.10	COMPONENT_REFERENCE		YES	—
7.36.1		CLASSIFICATION	YES	—
7.36.2		COMPONENT_PARENT	YES	—
7.36.3		COMPONENT_PATH	YES	—
7.2.2.2		DEVICE_REVISION	YES	—
7.2.2.3		DEVICE_TYPE	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.6		MANUFACTURER	YES	—
7.36.13		PROTOCOL	YES	—
7.11	COMPONENT_RELATION		YES	—
7.11.2.1		COMPONENTS	YES	—
7.36.9		LABEL	YES	—
7.36.8		HELP	YES	—
7.11.2.2		RELATION_TYPE	YES	—
7.11.2.3		ADDRESSING	YES	—
7.11.2.4		MAXIMUM_NUMBER	YES	—
7.11.2.5		MINIMUM_NUMBER	YES	—
7.11.2.6		REQUIRED_INTERFACE	NO	—
7.36.15		SUPPLIED_INTERFACE	NO	—
7.14	EDIT_DISPLAY		NO	—
7.15	FILE		YES	—
7.36.12		MEMBERS	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.6		HANDLING	YES	—
7.36.21		IDENTITY	YES	—
7.15.2.1		SHARED	YES	—
7.15.2.2		ON_UPDATE_ACTIONS	NO	—
7.16	GRAPH		YES	—
7.36.12		MEMBERS	YES	—
7.16.2.1		CYCLE_TIME	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.16.2.2		X_AXIS	YES	—
7.17	GRID		YES	—
7.17.2.1		VECTORS	YES	—
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.17.2.2		ORIENTATION	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.18	IMAGE		YES	—
7.18.2.1		PATH	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.18.2.2		LINK	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.19	IMPORT		YES	—
7.20	INTERFACE		NO	—
7.21	LIKE		YES	—
7.22	LIST		YES	—
7.22.2.1		TYPE	YES	—
7.22.2.2		CAPACITY	YES	—
7.22.2.3		COUNT	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.23	MENU		YES	—
7.23.2.1		ITEMS	YES	—
7.36.9		LABEL	YES	—
7.23.2.2		ACCESS	YES	—
7.23.2.12		EXIT_ACTIONS	YES	—
7.36.8		HELP	YES	—
7.23.2.13		INIT_ACTIONS	YES	—
7.23.2.3		POST_EDIT_ACTIONS	YES	—
7.23.2.4		POST_READ_ACTIONS	YES	—
7.23.2.5		POST_WRITE_ACTIONS	YES	—
7.23.2.6		PRE_EDIT_ACTIONS	YES	—
7.23.2.7		PRE_READ_ACTIONS	YES	—
7.23.2.8		PRE_WRITE_ACTIONS	YES	—
7.23.2.11		STYLE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.24	METHOD		YES	—
7.24.1		method_parameter	YES	—
7.36.1		DEFINITION	YES	—
7.24.2.1		ACCESS	YES	—
7.24.2.2		CLASS	YES	—
7.36.9		LABEL	YES	—
7.36.8		HELP	YES	—
7.36.18		PRIVATE	YES	—
7.24.2.3		TYPE	YES	—
7.36.16		VALIDITY	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.19		VISIBILITY	YES	—
7.42	PLUGIN		YES	—
7.36.9		LABEL	YES	—
7.36.8		HELP	YES	—
7.42.2		UUID	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.26	RECORD		NO	—
7.27	REFERENCE_ARRAY		YES	—
7.6.2		item-type	YES	Only item types supported by this communication profile.
7.27.2		ELEMENTS	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.28.1	REFRESH		YES	—
7.28.2	UNIT		YES	—
7.28.3	WRITE_AS_ONE		YES	—
7.36.14	RESPONSE_CODES		YES	—
7.30	SOURCE		YES	—
7.36.12		MEMBERS	YES	—
7.36.5		EMPHASIS	YES	—
7.30.2.1		EXIT_ACTIONS	YES	—
7.36.8		HELP	YES	—
7.30.2.2		INIT_ACTIONS	YES	—
7.36.9		LABEL	YES	—
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—
7.30.2.3		REFRESH_ACTIONS	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.30.2.4		Y_AXIS	YES	—
7.31	TEMPLATE		YES	—
7.31.2		DEFAULT_VALUES	YES	—
7.36.8		HELP	YES	—
7.36.9		LABEL	YES	—
7.36.16		VALIDITY	YES	—
7.32	VALUE_ARRAY		YES	—
7.36.9		LABEL	YES	—
7.32.2.1		NUMBER_OF_ELEMENTS	YES	—
7.32.2.2		TYPE	YES	—
7.36.8		HELP	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36.14		RESPONSE_CODES	YES	—
7.36.18		PRIVATE	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.20		WRITE_MODE	NO	—
7.33	VARIABLE		YES	—
7.33.2.1		CLASS	YES	—
7.36.9		LABEL	YES	—
7.33.2.2		TYPE	YES	—
7.33.2.3		CONSTANT_UNIT	YES	—
7.33.2.4		DEFAULT_VALUE	YES	—
7.36.6		HANDLING	YES	—
7.36.7		HEIGHT	YES	—
7.36.8		HELP	YES	—
7.33.2.5		INITIAL_VALUE	YES	—
7.33.2.6		POST_EDIT_ACTIONS	YES	—
7.33.2.7		POST_READ_ACTIONS	YES	—
7.33.2.15		POST_READ_UPDATE_ACTIONS	NO	—
7.33.2.16		POST_USERCHANGE_ACTIONS	NO	—
7.33.2.8		POST_WRITE_ACTIONS	YES	—
7.33.2.9		PRE_EDIT_ACTIONS	YES	—
7.33.2.10		PRE_READ_ACTIONS	YES	—
7.33.2.11		PRE_WRITE_ACTIONS	YES	—
7.36.18		PRIVATE	YES	—
7.33.2.13		REFRESH_ACTIONS	YES	—
7.36.14		RESPONSE_CODES	NO	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.36.17		WIDTH	YES	—
7.36.20		WRITE_MODE	NO	—
7.33.2.15	VARIABLE_LIST		NO	—
7.35	WAVEFORM		YES	—
7.35.2.1		TYPE	YES	—
7.36.5		EMPHASIS	YES	—
7.35.2.2		EXIT_ACTIONS	YES	—
7.36.6		HANDLING	YES	—
7.36.8		HELP	YES	—
7.35.2.3		INIT_ACTIONS	YES	—
7.35.2.4		KEY_POINTS	YES	—
7.36.9		LABEL	YES	—
7.36.10		LINE_COLOR	YES	—
7.36.11		LINE_TYPE	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.35.2.5		REFRESH_ACTIONS	YES	—
7.36.16		VALIDITY	YES	—
7.36.19		VISIBILITY	YES	—
7.35.2.6		Y_AXIS	YES	—
7.36.21	Conditional expressions		YES	—
7.38	Referencing		YES	—
7.39	Strings		YES	—
7.40	Expressions		YES	—
7.41	Text dictionary		YES	—
7.33.2.2.2.2	DATE_AND_TIME		YES	—
7.33.2.2.2.8	BOOLEAN		YES	Coding: TRUE is represented by the value 0xFF
NOTE Text is only ISO Latin-1.				

D.5.2 Builtin profile

The Builtin profiles are specified in IEC 61804-5.

D.5.3 EDDL Formal Definition profile

Reference	Title	Presence	Constraints
A.1	EDDL preprocessor	YES	—
A.2	Conventions	YES	—
A.3	Operators	YES	—
A.4	Keywords	YES	—
A.5	Terminals	YES	—
A.6	Formal EDDL syntax	YES	—

D.6 Data types

D.6.1 METHOD DEFINITION data types

Table D.8 specifies the METHOD DEFINITION data types.

Table D.8 – METHOD DEFINITION data types

data types	Minimum value	Maximum value	Remark
char	-128	+127	—
char[size]	-128 for each character	+127 for each character	—
char[]	-128 for each character	+127 for each character	Unspecified length of a character string.
short	-32 768	+32 767	—
int/ long	-2 147 483 648	+2 147 483 647	—
long long	-9 223 372 036 854 775 808	+9 223 372 036 854 775 807	—
unsigned char	0	+255	—
unsigned short	0	+65 535	—
unsigned int / unsigned long	0	+4 294 967 295	—
unsigned long long	0	+18 446 744 073 709 551 615	—
float	-3,402 823 466 × 10 ³⁸	+3,402 823 466 × 10 ³⁸	—
double	-1,797 693 134 862 315 7 × 10 ³⁰⁸	+1,797 693 134 862 315 7 × 10 ³⁰⁸	—
time_t	Not specified	Not specified	Opaque data type whose internal structure is specified by EDD application.
DD_STRING	—	—	Arbitrary length character string.
DD_ITEM	0	+4 294 967 295	ID of any item in the EDD.

D.6.2 VARIABLE TYPE data types**D.6.2.1 General**

Table D.9 specifies the VARIABLE TYPE data types. The specified size in brackets of the VARIABLE TYPES ensures that an overflow or underflow cannot occur.

Table D.9 – VARIABLE TYPES

VARIABLE TYPES	Minimum value	Maximum value	Initial value
BOOLEAN	See D.6.2.8	See D.6.2.8	FALSE
DOUBLE	$-1,797\ 693\ 134\ 862\ 315\ 7 \times 10^{308}$	$+1,797\ 693\ 134\ 862\ 315\ 7 \times 10^{308}$	0,0
FLOAT	$-3,402\ 823\ 466 \times 10^{38}$	$+3,402\ 823\ 466 \times 10^{38}$	0,0
INTEGER(1)	-128	+127	1
INTEGER(2)	-32 768	+32 767	1
INTEGER(3)	-8 388 608	+8 388 607	1
INTEGER(4)	-2 147 483 648	+2 147 483 647	1
INTEGER(5)	-549 755 813 888	+549 755 813 887	1
INTEGER(6)	-140 737 488 355 328	+140 737 488 355 327	1
INTEGER(7)	-36 028 797 018 963 968	+36 028 797 018 963 967	1
INTEGER(8)	-9 223 372 036 854 775 808	+9 223 372 036 854 775 807	1
UNSIGNED_INTEGER(1)	0	+255	1
UNSIGNED_INTEGER(2)	0	+65 535	1
UNSIGNED_INTEGER(3)	0	+16 777 215	1
UNSIGNED_INTEGER(4)	0	+4 294 967 295	1
UNSIGNED_INTEGER(5)	0	1 099 511 627 775	1
UNSIGNED_INTEGER(6)	0	281 474 976 710 655	1
UNSIGNED_INTEGER(7)	0	+72 057 594 037 927 935	1
UNSIGNED_INTEGER(8)	0	+18 446 744 073 709 551 615	1
DATE	See D.6.2.2	See D.6.2.2	1-Jan-1900
DATE_AND_TIME	See D.6.2.3	See D.6.2.3	1-Jan-1900 00:00:00
DURATION	See D.6.2.4	See D.6.2.4	0 days, 0 hours, 0 minutes, 0 seconds
TIME	See D.6.2.5	See D.6.2.5	00:00:00
TIME_VALUE(4)	See D.6.2.6	See D.6.2.6	00:00:00
TIME_VALUE(8)	See D.6.2.6	See D.6.2.6	1-Jan-1972 00:00:00
BIT_ENUMERATED(1)	0	+255	0
BIT_ENUMERATED(2)	0	+65 535	0
BIT_ENUMERATED(3)	0	+16 777 215	0
BIT_ENUMERATED(4)	0	+4 294 967 295	0
BIT_ENUMERATED(5)	0	1 099 511 627 775	0
BIT_ENUMERATED(6)	0	281 474 976 710 655	0
BIT_ENUMERATED(7)	0	+72 057 594 037 927 935	0
BIT_ENUMERATED(8)	0	+18 446 744 073 709 551 615	0
ENUMERATED(1)	0	+255	The value of the first enumeration.
ENUMERATED(2)	0	+65 535	The value of the first enumeration.
ENUMERATED(3)	0	+16 777 215	The value of the first enumeration.
ENUMERATED(4)	0	+4 294 967 295	The value of the first enumeration.

VARIABLE TYPES	Minimum value	Maximum value	Initial value
ENUMERATED(5)	0	1 099 511 627 775	The value of the first enumeration.
ENUMERATED(6)	0	281 474 976 710 655	The value of the first enumeration.
ENUMERATED(7)	0	+72 057 594 037 927 935	The value of the first enumeration.
ENUMERATED(8)	0	+18 446 744 073 709 551 615	The value of the first enumeration.
INDEX(1)	0	+255	First element in the associated reference array.
INDEX(2)	0	+65 535	First element in the associated reference array.
INDEX(3)	0	+16 777 215	First element in the associated reference array.
INDEX(4)	0	+4 294 967 295	First element in the associated reference array.
OBJECT_REFERENCE	N.a.	N.a.	SELF
ASCII(size)	0	+255 per character	Each character set to space.
BITSTRING(size)	0	1 per bit	Each bit set to 0.
EUC(size)	0	+255 per character	Each character set to space.
OCTET(size)	0	+255 per character	Each character set to space.
PACKED_ASCII(size)	See D.6.2.7	See D.6.2.7	Each character set to question mark (?).
PASSWORD(size)	0	+255 per character	Each character set to space.
VISIBLE(size)	0	+255 per character	Each character set to space.
Key:			
N.a.: not applicable			
NOTE The initial value may depend on the communication profile (see Table D.4, Table D.5, Table D.6 and Table D.7).			

D.6.2.2 Coding of DATE

Table D.10 specifies the DATE coding.

Table D.10 – DATE coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	0	0	0	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	1..31 day of month
2	0	0	0	0	2 ³	2 ²	2 ¹	2 ⁰	1...12 month
3	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	0...255 years (actual year minus 1900)

D.6.2.3 Coding of DATE_AND_TIME

Table D.11 specifies the DATE_AND_TIME coding.

Table D.11 – DATE_AND_TIME coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	0...59 999 ms
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
3	R		2^5	2^4	2^3	2^2	2^1	2^0	0...59 min
4	S	R		2^4	2^3	2^2	2^1	2^0	0...23 h
5	2^2	2^1	2^0	2^4	2^3	2^2	2^1	2^0	Bits 6 to 8: 1..7 day of week (1 = Monday, 7 = Sunday) Bits 1 to 5: 1..31 day of month
6	R		0	0	2^3	2^2	2^1	2^0	1...12 months
7	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0...255 years (actual year minus 1900) NOTE The coding of the year depends on the communication profile (see Table D.4, Table D.5, Table D.6 and Table D.7).
R means reserved for future use; S = 0 indicates standard time, S = 1 indicates summer time.									

D.6.2.4 Coding of DURATION

Table D.12 specifies the DURATION coding.

Table D.12 – DURATION coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of ms since midnight.
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Number of days.
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

D.6.2.5 Coding of TIME

Table D.13 specifies the TIME coding.

Table D.13 – TIME coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds since midnight.
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Number of days since 01.01.1984.
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

D.6.2.6 Coding of TIME_VALUE

Table D.14 specifies the four-octet TIME_VALUE coding.

Table D.14 – TIME_VALUE coding (four octets)

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Time difference or time of day since midnight as positive number of 1/32 ms.
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Table D.15 specifies the eight octet TIME_VALUE coding.

Table D.15 – TIME_VALUE coding (eight octets)

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	0	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}	Absolute time as positive number of 1/32 ms since January 1, 1972.
2	2^{65}	2^{64}	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}	
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}	
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

D.6.2.7 Coding of PACKED_ASCII (6-BIT ASCII) DATA FORMAT

Some of the alpha-numeric data passed by the protocol is transmitted to and from the devices in the PACKED_ASCII format. PACKED_ASCII is a subset of ASCII produced by removing the two most significant bits of each ASCII character. This allows four PACKED_ASCII characters to be placed in the space of three ASCII characters. Typically, PACKED_ASCII strings are defined in even multiples of three octets, four ASCII characters.

Construction of PACKED_ASCII characters:

- Truncate Bit #6 and #7 of each ASCII character.
- Pack four, 6-bit ASCII characters into three octets.

Reconstruction of ASCII characters:

- Unpack the four, 6-bit ASCII characters.
- Place the complement of Bit #5 of each unpacked, 6-bit ASCII character into Bit #6.
- Set Bit #7 of each of the unpacked ASCII characters to zero.

Table D.16 shows the coding of the PACKED_ASCII characters.

Table D.16 – PACKED_ASCII coding

Character	Code	Character	Code	Character	Code	Character	Code
@	00	P	10	Space	20	0	30
A	01	Q	11	!	21	1	31
B	02	R	12	"	22	2	32
C	03	S	13	#	23	3	33
D	04	T	14	\$	24	4	34
E	05	U	15	%	25	5	35
F	06	V	16	&	26	6	36
G	07	W	17	'	27	7	37
H	08	X	18	(	28	8	38
I	09	Y	19	)	29	9	39
J	0A	Z	1A	*	2A	:	3A
K	0B	[	1B	+	2B	;	3B
L	0C	\	1C	,	2C	<	3C
M	0D	]	1D	-	2D	=	3D
N	0E	^	1E	.	2E	>	3E
O	0F	_	1F	/	2F	?	3F

D.6.2.8 Coding of BOOLEAN

A BOOLEAN value is a one-byte unsigned integer. Table D.17 specifies the BOOLEAN coding.

Table D.17 – BOOLEAN coding

Value	Code	Description
FALSE	0	
TRUE	any other value	It is recommended not to compare with TRUE in conditions. Instead compare with FALSE, e.g. 'a != FALSE'. NOTE The coding of TRUE depends on the communication profile (see Table D.4, Table D.5, Table D.6 and Table D.7).

Bibliography

IEC 60050-351, *International Electrotechnical Vocabulary – Part 351: Control technology*

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

IEC 61499-1, *Function blocks – Part 1: Architecture*

IEC 61784-1, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61784-2, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

IEC 61804-4⁸, *Function blocks (FB) for process control and Electronic Device Description Language (EDDL) – Part 4: EDD interpretation*

ISO/IEC 2022, *Information technology – Character code structure and extension techniques*

ISO 2382 (all parts), *Information technology – Vocabulary*

ISO 15745-1, *Industrial automation systems and integration – Open systems application integration framework – Part 1: Generic reference description*

ISO/AFNOR, *Dictionary of computer science – The standardized vocabulary*

⁸ To be published.

IECNORM.COM :: Click to view the full PDF of IEC 61804-3:2015

SOMMAIRE

AVANT-PROPOS.....	356
INTRODUCTION.....	358
1 Domaine d'application.....	360
2 Références normatives.....	360
3 Termes, définitions, termes abrégés et acronymes	361
3.1 Termes et définitions.....	361
3.2 Termes abrégés et acronymes.....	363
4 Déclaration de conformité	364
5 Conventions	364
5.1 Généralités	364
5.2 Conventions pour la structure lexicale	364
5.2.1 ABC champ1, champ2.....	364
5.2.2 ABC champ1+.....	365
5.2.3 ABC champ2*	365
5.2.4 ABC [champ1, champ2]+.....	365
5.2.5 ABC champ1, (champ2, champ3)<exp>	365
6 Modèle EDD et EDDL	366
6.1 Présentation des EDD et du langage EDDL	366
6.2 Architecture EDD	366
6.3 Concepts d'EDD.....	366
6.4 Principes du processus de développement d'EDD	367
6.4.1 Généralités.....	367
6.4.2 Génération du code source de l'EDD	367
6.4.3 Prétraitement de l'EDD.....	367
6.4.4 Compilation de l'EDD.....	368
6.5 Interrelations entre la structure lexicale et les définitions formelles.....	368
6.6 Builtins.....	368
6.7 Profils.....	368
7 Langage de description électronique de produit (EDDL).....	368
7.1 Vue d'ensemble	368
7.1.1 Caractéristiques du langage EDDL	368
7.1.2 Représentation de syntaxe	369
7.1.3 Eléments de langage d'une EDD	369
7.1.4 Eléments de construction de base	369
7.1.5 Attributs communs	380
7.1.6 Eléments spéciaux.....	380
7.1.7 Règles pour les instances	381
7.1.8 Règles pour une liste d'attributs VARIABLE	381
7.2 Informations d'identification d'une EDD.....	381
7.2.1 Structure générale	381
7.2.2 Attributs spécifiques.....	382
7.3 AXIS.....	385
7.3.1 Structure générale	385
7.3.2 Attributs spécifiques.....	385
7.4 BLOCK	387
7.4.1 BLOCK_A	387

7.4.2	BLOCK_B	399
7.5	CHART	401
7.5.1	Structure générale	401
7.5.2	Attributs spécifiques	402
7.6	COLLECTION	403
7.6.1	Structure générale	403
7.6.2	Attribut spécifique – item-type	404
7.7	COMMAND	405
7.7.1	Structure générale	405
7.7.2	Attributs spécifiques	406
7.8	COMPONENT	412
7.8.1	Structure générale	412
7.8.2	Attributs spécifiques	413
7.9	COMPONENT_FOLDER	418
7.10	COMPONENT_REFERENCE	419
7.11	COMPONENT_RELATION	420
7.11.1	Structure générale	420
7.11.2	Attributs spécifiques	420
7.12	CONNECTION (vide)	423
7.13	DOMAIN (vide)	423
7.14	EDIT_DISPLAY	423
7.14.1	Structure générale	423
7.14.2	Attributs spécifiques	424
7.15	FILE	426
7.15.1	Structure générale	426
7.15.2	Attributs spécifiques	427
7.16	GRAPH	428
7.16.1	Structure générale	428
7.16.2	Attributs spécifiques	428
7.17	GRID	429
7.17.1	Structure générale	429
7.17.2	Attributs spécifiques	430
7.18	IMAGE	431
7.18.1	Structure générale	431
7.18.2	Attributs spécifiques	431
7.19	IMPORT	432
7.19.1	Structure générale	432
7.19.2	Redéfinitions	435
7.20	INTERFACE	452
7.20.1	Structure générale	452
7.20.2	Attribut spécifique – DECLARATION	452
7.21	LIKE	453
7.22	LIST	453
7.22.1	Structure générale	453
7.22.2	Attributs spécifiques	454
7.23	MENU	455
7.23.1	Structure générale	455
7.23.2	Attributs spécifiques	456
7.23.3	Schémas de séquence pour les actions	462

7.24	METHOD	464
7.24.1	Structure générale	464
7.24.2	Attributs spécifiques	465
7.25	PROGRAM (vide)	467
7.26	RECORD	467
7.27	REFERENCE_ARRAY	468
7.27.1	Structure générale	468
7.27.2	Attribut spécifique – ELEMENTS	468
7.28	Relations	469
7.28.1	REFRESH	469
7.28.2	UNIT	470
7.28.3	WRITE_AS_ONE	470
7.29	RESPONSE_CODES	471
7.30	SOURCE	471
7.30.1	Structure générale	471
7.30.2	Attributs spécifiques	472
7.31	TEMPLATE	473
7.31.1	Structure générale	473
7.31.2	Attribut spécifique – DEFAULT_VALUES	474
7.32	VALUE_ARRAY	474
7.32.1	Structure générale	474
7.32.2	Attributs spécifiques	475
7.33	VARIABLE	476
7.33.1	Structure générale	476
7.33.2	Attributs spécifiques	477
7.34	VARIABLE_LIST	496
7.35	WAVEFORM	497
7.35.1	Structure générale	497
7.35.2	Attributs spécifiques	498
7.36	Attributs communs	504
7.36.1	CLASSIFICATION	504
7.36.2	COMPONENT_PARENT	506
7.36.3	COMPONENT_PATH	506
7.36.4	DEFINITION	507
7.36.5	EMPHASIS	507
7.36.6	HANDLING	508
7.36.7	HEIGHT	508
7.36.8	HELP	509
7.36.9	LABEL	510
7.36.10	LINE_COLOR	510
7.36.11	LINE_TYPE	510
7.36.12	MEMBERS	511
7.36.13	PROTOCOL	512
7.36.14	RESPONSE_CODES	513
7.36.15	SUPPLIED_INTERFACE	513
7.36.16	VALIDITY	514
7.36.17	WIDTH	514
7.36.18	PRIVATE	514
7.36.19	VISIBILITY	515

7.36.20	WRITE_MODE	515
7.36.21	IDENTITY	516
7.37	Expression conditionnelle	516
7.38	Référencement	517
7.38.1	Référencement d'une instance d'EDD	517
7.38.2	Référencement de bits de BIT_ENUMERATED VARIABLE	518
7.38.3	Référencement des membres d'un RECORD	518
7.38.4	Référencement d'éléments d'un attribut VALUE_ARRAY	518
7.38.5	Référencement des membres d'une COLLECTION	519
7.38.6	Référencement d'éléments d'un attribut REFERENCE_ARRAY	519
7.38.7	Référencement des membres d'un attribut VARIABLE_LISTS	520
7.38.8	Référencement d'éléments BLOCK_A PARAMETERS	520
7.38.9	Référencement d'éléments de BLOCK_A PARAMETER_LISTS	520
7.38.10	Référencement d'éléments de BLOCK_A LOCAL_PARAMETERS	521
7.38.11	Référencement de BLOCK_A CHARACTERISTICS	521
7.38.12	Référencement des membres d'un attribut FILE	521
7.38.13	Référencement d'éléments de LIST	522
7.38.14	Référencement des membres de CHART	522
7.38.15	Référencement de membres d'un attribut GRAPH	522
7.38.16	Référencement des membres de SOURCE	523
7.38.17	Référencement d'AXIS de GRAPH, SOURCE, WAVEFORM	523
7.38.18	Référencement de PARAMETERS d'une instance spécifique de BLOCK_A	523
7.38.19	Référencement de LOCAL_PARAMETERS d'une instance spécifique de BLOCK_A	524
7.38.20	Référencement de CHARACTERISTICS d'une instance spécifique de BLOCK_A	524
7.38.21	Référencement de CHARTS d'une instance spécifique de BLOCK_A	525
7.38.22	Référencement de LISTS d'une instance spécifique de BLOCK_A	525
7.38.23	Référencement de GRAPHS d'une instance spécifique de BLOCK_A	526
7.38.24	Référencement de GRIDS d'une instance spécifique de BLOCK_A	526
7.38.25	Référencement de MENUS d'une instance spécifique de BLOCK_A	527
7.38.26	Référencement d'attributs METHODS d'une instance spécifique de BLOCK_A	527
7.38.27	Référencement des instances de COMPONENT	528
7.38.28	Référencement de types de COMPONENT	528
7.38.29	Référencement de FILES d'une instance spécifique de BLOCK_A	529
7.38.30	Référencement de PLUGINS d'une instance spécifique de BLOCK_A	529
7.39	Chaînes	530
7.39.1	Spécification d'une chaîne sous forme d'un littéral de chaîne	530
7.39.2	Spécification d'une chaîne en tant que variable de chaîne	530
7.39.3	Spécification d'une chaîne en tant que valeur d'énumération	530
7.39.4	Spécification d'une chaîne en tant que référence de dictionnaire	531
7.39.5	Référencement d'attributs HELP et LABEL d'instances d'EDD	531
7.39.6	Opérations de chaîne	532
7.39.7	Formats de chaîne d'invite de commande	532
7.40	Expression	533
7.40.1	Structure générale	533
7.40.2	Expressions primaires	533
7.40.3	Expressions unaires	536

7.40.4	Expressions binaires	537
7.41	Dictionnaire de texte	539
7.42	PLUGIN	540
7.42.1	Structure générale	540
7.42.2	Attribut spécifique – UUID	540
7.43	BLOB	541
Annexe A (normative)	Définition formelle d'EDDL	542
A.1	Préprocesseur EDDL	542
A.1.1	Structure générale	542
A.1.2	Directives	542
A.1.3	Macros prédéfinies	545
A.1.4	Caractères NEWLINE	546
A.1.5	Commentaires	546
A.2	Conventions	546
A.2.1	Constantes entières	546
A.2.2	Constantes à virgule flottante	546
A.2.3	Littéraux de chaîne	547
A.2.4	Utilisation de codes de langue et de pays dans des littéraux de chaîne	547
A.3	Opérateurs	548
A.4	Mots clés	552
A.5	Terminaux	556
A.6	Syntaxe EDDL formelle	557
A.6.1	Généralités	557
A.6.2	Informations d'identification d'une EDD	557
A.6.3	AXIS	559
A.6.4	BLOCK_A et BLOCK_B	559
A.6.5	CHART	564
A.6.6	COLLECTION	565
A.6.7	COMMAND	565
A.6.8	COMPONENT	568
A.6.9	COMPONENT_FOLDER	571
A.6.10	COMPONENT_REFERENCE	572
A.6.11	COMPONENT_RELATION	572
A.6.12	CONNECTION (vide)	574
A.6.13	DOMAIN (vide)	574
A.6.14	EDIT_DISPLAY	574
A.6.15	FILE	575
A.6.16	GRAPH	576
A.6.17	GRID	576
A.6.18	IMAGE	577
A.6.19	INTERFACE	578
A.6.20	LIST	578
A.6.21	IMPORT	579
A.6.22	LIKE	581
A.6.23	MENU	582
A.6.24	METHOD	584
A.6.25	PROGRAM (vide)	586
A.6.26	RECORD	586
A.6.27	REFERENCE_ARRAY	586

A.6.28	RELATIONS.....	587
A.6.29	RESPONSE_CODES	589
A.6.30	SOURCE	589
A.6.31	TEMPLATE	590
A.6.32	VALUE_ARRAY	590
A.6.33	VARIABLE	591
A.6.34	VARIABLE_LIST	600
A.6.35	WAVEFORM.....	600
A.6.36	Attributs communs	602
A.6.37	Expression.....	606
A.6.38	Syntaxe C.....	609
A.6.39	Redéfinition	612
A.6.40	Références	636
A.6.41	PLUGIN	639
A.6.42	BLOB.....	639
A.7	Syntaxe de dictionnaire formelle	640
Annexe B (normative)	Bibliothèque de Builtin EDDL (vide).....	641
Annexe C (informative)	Exemple d'EDD.....	642
C.1	Exemple d'EDD d'un transmetteur de température.....	642
C.2	Exemple d'EDD.....	643
Annexe D (normative)	Profils d'EDDL et de Builtins	656
D.1	Conventions pour les profils d'EDDL et de Builtins	656
D.2	Profils pour PROFIBUS et PROFINET.....	657
D.2.1	Profil EDDL.....	657
D.2.2	Profil de Builtin	663
D.2.3	Profil de définition formelle d'EDDL	664
D.3	Profils pour FOUNDATION™ Fieldbus	664
D.3.1	Profil EDDL.....	664
D.3.2	Profil de Builtin	671
D.3.3	Profil de définition formelle d'EDDL	671
D.4	Profils pour HART® Communication Foundation (HCF)	671
D.4.1	Profil EDDL.....	671
D.4.2	Profil de Builtin	679
D.4.3	Profil de définition formelle d'EDDL	679
D.5	Profils pour les serveurs de communication	679
D.5.1	Profil EDDL.....	679
D.5.2	Profil de Builtin	686
D.5.3	Profil de définition formelle d'EDDL	686
D.6	Types de données.....	686
D.6.1	Types de données de DEFINITION de METHOD.....	686
D.6.2	Types de données de TYPE de VARIABLE	687
Bibliographie		694
Figure 1	– Position de l'IEC 61804 par rapport à d'autres normes et produits	359
Figure 2	– Processus de génération d'EDD.....	367
Figure 3	– BLOCK_A	370
Figure 4	– CHART	370
Figure 5	– COLLECTION	371

Figure 6 – COMMAND	371
Figure 7 – COMPONENT	372
Figure 8 – COMPONENT_FOLDER	372
Figure 9 – COMPONENT_REFERENCE	372
Figure 10 – COMPONENT_RELATION	373
Figure 11 – EDIT_DISPLAY	373
Figure 12 – FILE	373
Figure 13 – GRAPH	374
Figure 14 – GRID	374
Figure 15 – IMAGE	374
Figure 16 – LIKE	375
Figure 17 – LIST	375
Figure 18 – MENU	376
Figure 19 – RECORD	376
Figure 20 – REFERENCE_ARRAY	377
Figure 21 – REFRESH	377
Figure 22 – UNIT	378
Figure 23 – WRITE_AS_ONE	378
Figure 24 – SOURCE	378
Figure 25 – VALUE_ARRAY	379
Figure 26 – VARIABLE	379
Figure 27 – VARIABLE_LIST	379
Figure 28 – WAVEFORM	380
Figure 29 – Mécanismes d'importation EDDL	433
Figure 30 – Activation de MENU	464
Figure C.1 – Exemple d'écran d'opérateur pour l'utilisation d'une EDD	642
Tableau 1 – Descriptions d'attributs de champ	365
Tableau 2 – Attribut DD_REVISION	382
Tableau 3 – Attribut DEVICE_REVISION	382
Tableau 4 – Attributs DEVICE_TYPE	383
Tableau 5 – Attribut EDD_PROFILE	383
Tableau 6 – Attribut EDD_VERSION	384
Tableau 7 – Attributs MANUFACTURER	384
Tableau 8 – Attribut MANUFACTURER_EXT	384
Tableau 9 – Attributs AXIS	385
Tableau 10 – Attributs MAX_VALUE, MIN_VALUE	386
Tableau 11 – Attributs SCALING	386
Tableau 12 – Attributs BLOCK_A	388
Tableau 13 – Attribut CHARACTERISTICS	389
Tableau 14 – Attributs PARAMETERS	389
Tableau 15 – Attribut AXIS_ITEMS	390
Tableau 16 – Attribut CHART_ITEMS	390

Tableau 17 – Attribut COLLECTION_ITEMS	390
Tableau 18 – Attribut EDIT_DISPLAY_ITEMS.....	391
Tableau 19 – Attribut FILE_ITEMS	391
Tableau 20 – Attribut GRAPH_ITEMS	391
Tableau 21 – Attribut GRID_ITEMS	392
Tableau 22 – Attribut IMAGE_ITEMS.....	392
Tableau 23 – Attribut LIST_ITEMS	392
Tableau 24 – Attribut MENU_ITEMS.....	393
Tableau 25 – Attribut METHOD_ITEMS	393
Tableau 26 – Attributs PARAMETER_LISTS.....	394
Tableau 27 – Attribut REFERENCE_ARRAY_ITEMS	394
Tableau 28 – Attribut REFRESH_ITEMS	394
Tableau 29 – Attribut SOURCE_ITEMS	395
Tableau 30 – Attribut UNIT_ITEMS	395
Tableau 31 – Attribut WAVEFORM_ITEMS.....	395
Tableau 32 – Attribut WRITE_AS_ONE_ITEMS	396
Tableau 33 – Attributs CHARTS	396
Tableau 34 – Attributs LISTS	396
Tableau 35 – Attributs GRAPHS.....	397
Tableau 36 – Attributs GRIDS	397
Tableau 37 – Attributs MENUS.....	398
Tableau 38 – Attributs METHODS	398
Tableau 39 – Attributs FILES	399
Tableau 40 – Attribut PLUGIN_ITEMS.....	399
Tableau 41 – Attributs PLUGINS.....	399
Tableau 42 – Attributs BLOCK_B	400
Tableau 43 – Attributs NUMBER	400
Tableau 44 – Attributs TYPE.....	401
Tableau 45 – Attributs CHART	402
Tableau 46 – Attribut CYCLE_TIME	402
Tableau 47 – Attribut LENGTH.....	403
Tableau 48 – Attributs TYPE	403
Tableau 49 – Attributs COLLECTION	404
Tableau 50 – item-type	405
Tableau 51 – Attributs COMMAND	406
Tableau 52 – Attributs OPERATION	406
Tableau 53 – Attributs TRANSACTION.....	407
Tableau 54 – Attributs REPLY et REQUEST	408
Tableau 55 – Attributs INDEX	409
Tableau 56 – Attribut BLOCK_B	409
Tableau 57 – Attribut NUMBER	410
Tableau 58 – Attributs SLOT	410
Tableau 59 – Attributs SUB_SLOT	411

Tableau 60 – Attribut HEADER.....	411
Tableau 61 – Attributs API	412
Tableau 62 – Attribut POST_RQSTRECEIVE_ACTIONS.....	412
Tableau 63 – Attributs COMPONENT	413
Tableau 64 – Attributs CAN_DELETE.....	414
Tableau 65 – Attribut CHECK_CONFIGURATION	414
Tableau 66 – Attribut COMPONENT_RELATIONS	414
Tableau 67 – Attribut DECLARATION.....	415
Tableau 68 – Attribut DETECT	415
Tableau 69 – Attribut EDD	416
Tableau 70 – Attribut INITIAL_VALUES.....	416
Tableau 71 – Attribut REDUNDANCY	416
Tableau 72 – Attribut SCAN	417
Tableau 73 – Attribut SCAN_LIST	417
Tableau 74 – Attributs BYTE_ORDER	418
Tableau 75 – Attribut CONNECTION_POINT	418
Tableau 76 – Attribut PRODUCT_URI	418
Tableau 77 – Attributs COMPONENT_FOLDER.....	419
Tableau 78 – Attributs COMPONENT_REFERENCE.....	420
Tableau 79 – Attributs COMPONENT_RELATION.....	420
Tableau 80 – Attributs COMPONENTS.....	421
Tableau 81 – Attributs RELATION_TYPE.....	422
Tableau 82 – Attribut ADDRESSING	422
Tableau 83 – Attribut MAXIMUM_NUMBER.....	423
Tableau 84 – Attribut MINIMUM_NUMBER.....	423
Tableau 85 – Attribut REQUIRED_INTERFACE.....	423
Tableau 86 – Attributs EDIT_DISPLAY	424
Tableau 87 – Attribut EDIT_ITEMS.....	424
Tableau 88 – Attribut DISPLAY_ITEM	425
Tableau 89 – Attributs POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS	426
Tableau 90 – Attributs FILE.....	427
Tableau 91 – Attributs SHARED.....	427
Tableau 92 – Attribut ON_UPDATE_ACTIONS	428
Tableau 93 – Attributs GRAPH.....	428
Tableau 94 – Attribut CYCLE_TIME	429
Tableau 95 – Attribut X_AXIS.....	429
Tableau 96 – Attributs GRID	430
Tableau 97 – Attributs VECTORS.....	430
Tableau 98 – Attributs ORIENTATION.....	431
Tableau 99 – Attributs IMAGE	431
Tableau 100 – Attribut PATH.....	432
Tableau 101 – Attribut LINK.....	432
Tableau 102 – Importation de description d'un appareil	434

Tableau 103 – Attributs de redéfinition	435
Tableau 104 – Règles de redéfinition pour les attributs AXIS	436
Tableau 105 – Règles de redéfinition pour les attributs BLOB	436
Tableau 106 – Règles de redéfinition pour les attributs BLOCK_A	437
Tableau 107 – Règles de redéfinition pour les attributs BLOCK_B	438
Tableau 108 – Règles de redéfinition pour les attributs CHART	438
Tableau 109 – Règles de redéfinition pour les attributs COLLECTION	439
Tableau 110 – Règles de redéfinition pour les attributs COMMAND	439
Tableau 111 – Règles de redéfinition pour les attributs COMPONENT	440
Tableau 112 – Règles de redéfinition pour les attributs COMPONENT_FOLDER	440
Tableau 113 – Règles de redéfinition pour les attributs COMPONENT_REFERENCE	441
Tableau 114 – Règles de redéfinition pour les attributs COMPONENT_RELATION	441
Tableau 115 – Règles de redéfinition pour les attributs EDIT_DISPLAY	442
Tableau 116 – Règles de redéfinition pour les attributs FILE	442
Tableau 117 – Règles de redéfinition pour les attributs GRAPH	443
Tableau 118 – Règles de redéfinition pour les attributs GRID	443
Tableau 119 – Règles de redéfinition pour les attributs IMAGE	444
Tableau 120 – Règles de redéfinition pour les attributs INTERFACE	444
Tableau 121 – Règles de redéfinition pour les attributs LIST	444
Tableau 122 – Règles de redéfinition pour les attributs MENU	445
Tableau 123 – Règles de redéfinition pour les attributs METHOD	446
Tableau 124 – Règles de redéfinition pour les attributs PLUGIN	446
Tableau 125 – Règles de redéfinition pour les attributs RECORD	447
Tableau 126 – Règles de redéfinition pour les attributs REFERENCE_ARRAY	447
Tableau 127 – Règles de redéfinition pour les attributs RESPONSE_CODES	447
Tableau 128 – Règles de redéfinition pour les attributs SOURCE	448
Tableau 129 – Règles de redéfinition pour les attributs TEMPLATE	448
Tableau 130 – Règles de redéfinition pour les attributs VALUE_ARRAY	449
Tableau 131 – Règles de redéfinition pour les attributs VARIABLE	450
Tableau 132 – Règles de redéfinition pour les attributs VARIABLE_LIST	451
Tableau 133 – Règles de redéfinition pour les attributs WAVEFORM	451
Tableau 134 – Attributs INTERFACE	452
Tableau 135 – Attributs DECLARATION	452
Tableau 136 – Attributs LIKE	453
Tableau 137 – Attributs LIST	453
Tableau 138 – Attribut TYPE	454
Tableau 139 – Attribut CAPACITY	454
Tableau 140 – Attribut COUNT	455
Tableau 141 – Attributs MENU	455
Tableau 142 – Attributs ITEMS	456
Tableau 143 – Attributs ACCESS	457
Tableau 144 – Attributs EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS	458

Tableau 145 – Attributs STYLE	461
Tableau 146 – Attributs METHOD	464
Tableau 147 – Types de paramètres	465
Tableau 148 – Attributs ACCESS	465
Tableau 149 – Attributs CLASS	466
Tableau 150 – Attributs TYPE	467
Tableau 151 – Attributs RECORD	467
Tableau 152 – Attributs REFERENCE_ARRAY	468
Tableau 153 – Attributs ELEMENTS	469
Tableau 154 – Attributs REFRESH	469
Tableau 155 – Attributs UNIT	470
Tableau 156 – Attribut WRITE_AS_ONE	470
Tableau 157 – Attributs RESPONSE_CODES	471
Tableau 158 – Attributs SOURCE	472
Tableau 159 – Attribut Y_AXIS	473
Tableau 160 – Attributs TEMPLATE	474
Tableau 161 – Attributs DEFAULT_VALUES	474
Tableau 162 – Attributs VALUE_ARRAY	475
Tableau 163 – Attributs NUMBER_OF_ELEMENTS	475
Tableau 164 – Attribut TYPE	476
Tableau 165 – Attributs VARIABLE	477
Tableau 166 – Attributs CLASS	478
Tableau 167 – Attributs TYPE	479
Tableau 168 – Attributs DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER	480
Tableau 169 – Attributs DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE	483
Tableau 170 – Attributs BIT_ENUMERATED	486
Tableau 171 – Attributs status-class	487
Tableau 172 – Attributs ALL, AO, DV, TV	488
Tableau 173 – Attributs de type ENUMERATED	488
Tableau 174 – Attributs de type INDEX	489
Tableau 175 – Attributs de types de chaînes	491
Tableau 176 – Attribut CONSTANT_UNIT	492
Tableau 177 – Attribut DEFAULT_VALUE	492
Tableau 178 – Attribut INITIAL_VALUE	493
Tableau 179 – Attributs POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS	493
Tableau 180 – Attribut POST_USERCHANGE_ACTIONS, POST_RQSTUPDATE_ACTIONS	496
Tableau 181 – Attributs VARIABLE_LIST	497
Tableau 182 – Attributs WAVEFORM	497
Tableau 183 – Attributs TYPE	498
Tableau 184 – Attributs XY	499
Tableau 185 – Attributs YT	500

Tableau 186 – Attribut HORIZONTAL	501
Tableau 187 – Attribut VERTICAL	501
Tableau 188 – Attributs EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS	502
Tableau 189 – Attributs KEY_POINTS	502
Tableau 190 – Attributs X_VALUES, Y_VALUES	503
Tableau 191 – Attribut Y_AXIS	504
Tableau 192 – Attributs CLASSIFICATION	505
Tableau 193 – Attribut COMPONENT_PARENT	506
Tableau 194 – Attribut COMPONENT_PATH	507
Tableau 195 – Attribut DEFINITION	507
Tableau 196 – Attributs EMPHASIS	508
Tableau 197 – Attributs HANDLING	508
Tableau 198 – Attributs HEIGHT/WIDTH	509
Tableau 199 – Attribut HELP	510
Tableau 200 – Attribut LABEL	510
Tableau 201 – Attributs LINE_COLOR	510
Tableau 202 – Attribut LINE_TYPE	511
Tableau 203 – Attributs MEMBERS	512
Tableau 204 – Attributs PROTOCOL	512
Tableau 205 – Attribut RESPONSE_CODES	513
Tableau 206 – Attribut SUPPLIED_INTERFACE	513
Tableau 207 – Attributs VALIDITY	514
Tableau 208 – Attributs PRIVATE	514
Tableau 209 – Attributs VISIBILITY	515
Tableau 210 – Attributs WRITE_MODE	515
Tableau 211 – Attribut IDENTITY	516
Tableau 212 – Instruction conditionnelle IF, SELECT	517
Tableau 213 – Référencement d'une instance d'EDD	518
Tableau 214 – Référencement d'éléments de VARIABLE	518
Tableau 215 – Référencement d'éléments de RECORD	518
Tableau 216 – Référencement d'éléments de VALUE_ARRAY	519
Tableau 217 – Référencement des membres de COLLECTION	519
Tableau 218 – Référencement des membres de REFERENCE_ARRAY	519
Tableau 219 – Référencement des membres de VARIABLE_LISTS	520
Tableau 220 – Référencement des membres de BLOCK_A PARAMETERS	520
Tableau 221 – Référencement des membres de BLOCK_A PARAMETER_LISTS	520
Tableau 222 – Référencement des membres de BLOCK_A LOCAL_PARAMETERS	521
Tableau 223 – Référencement de BLOCK_A CHARACTERISTICS	521
Tableau 224 – Référencement des membres de FILE	521
Tableau 225 – Référencement d'éléments de LIST	522
Tableau 226 – Référencement des membres de CHART	522
Tableau 227 – Référencement des membres de GRAPH	523
Tableau 228 – Référencement des membres de SOURCE	523

Tableau 229 – Référencement d'AXIS d'un attribut GRAPH, SOURCE, WAVEFORM.....	523
Tableau 230 – Référencement de PARAMETERS d'une instance spécifique de BLOCK_A.....	524
Tableau 231 – Référencement de LOCAL_PARAMETERS d'une instance spécifique de BLOCK_A.....	524
Tableau 232 – Référencement de CHARACTERISTICS d'une instance spécifique de BLOCK_A.....	525
Tableau 233 – Référencement de CHARTS d'une instance spécifique de BLOCK_A	525
Tableau 234 – Référencement de LISTS d'une instance spécifique de BLOCK_A.....	526
Tableau 235 – Référencement de GRAPHS d'une instance spécifique de BLOCK_A	526
Tableau 236 – Référencement de GRIDS d'une instance spécifique de BLOCK_A.....	527
Tableau 237 – Référencement de MENUS d'une instance spécifique de BLOCK_A.....	527
Tableau 238 – Référencement d'attributs METHODS d'une instance spécifique de BLOCK_A.....	528
Tableau 239 – Référencement d'une instance de COMPONENT.....	528
Tableau 240 – Référencement d'un type COMPONENT.....	529
Tableau 241 – Référencement de FILES d'une instance spécifique de BLOCK_A.....	529
Tableau 242 – Référencement de PLUGINS d'une instance spécifique de BLOCK_A	529
Tableau 243 – Chaîne sous forme de littéral de chaîne.....	530
Tableau 244 – Chaîne en tant que variable de chaîne.....	530
Tableau 245 – Chaîne en tant que valeur d'énumération.....	531
Tableau 246 – Chaîne en tant que référence de dictionnaire.....	531
Tableau 247 – Référencement des attributs HELP et LABEL des instances d'EDD	531
Tableau 248 – Opération de chaîne.....	532
Tableau 249 – Spécificateur de format.....	532
Tableau 250 – Expressions primaires.....	533
Tableau 251 – Valeurs d'attribut VARIABLE	535
Tableau 252 – Valeurs d'attribut AXIS.....	536
Tableau 253 – Valeurs d'attribut BLOB.....	536
Tableau 254 – Valeurs d'attribut LIST.....	536
Tableau 255 – Valeurs d'attribut ARRAY	536
Tableau 256 – Expressions unaires.....	536
Tableau 257 – Opérateurs multiplicatifs.....	537
Tableau 258 – Opérateurs additifs.....	537
Tableau 259 – Opérateurs de décalage	538
Tableau 260 – Opérateurs relationnels.....	538
Tableau 261 – Opérateurs d'égalité.....	538
Tableau 262 – Attributs de dictionnaire de texte	540
Tableau 263 – Attributs PLUGIN	540
Tableau 264 – Attribut UUID	541
Tableau 265 – Attributs BLOB.....	541
Tableau A.1 – Conventions pour les constantes entières	546
Tableau A.2 – Utilisation de séquences d'échappement dans des littéraux de chaîne	547
Tableau A.3 – Exemples de codes de langue pour les littéraux de chaîne	548

Tableau A.4 – Priorité et associativité pour les opérateurs EDDL	549
Tableau A.5 – Opérations pour les attributs VARIABLE ou les variables locales METHOD	551
Tableau A.6 – Mots clés EDDL	552
Tableau D.1 – Tableaux de sélection de profil	656
Tableau D.2 – Tableaux de profil de définition formelle d'EDDL	656
Tableau D.3 – Tableaux de sélection de contenu	656
Tableau D.4 – Sélection d'éléments EDDL pour PROFIBUS et PROFINET	657
Tableau D.5 – Sélection d'éléments EDDL pour FOUNDATION Fieldbus	664
Tableau D.6 – Sélection d'éléments EDDL pour HCF	671
Tableau D.7 – Sélection d'éléments EDDL pour les serveurs de communication	679
Tableau D.8 – Types de données de DEFINITION de METHOD	687
Tableau D.9 – Attributs TYPE de VARIABLE	688
Tableau D.10 – Codage de DATE	690
Tableau D.11 – Codage de DATE_AND_TIME	690
Tableau D.12 – Codage de DURATION	691
Tableau D.13 – Codage de TIME	691
Tableau D.14 – Codage de TIME_VALUE (quatre octets)	691
Tableau D.15 – Codage de TIME_VALUE (huit octets)	692
Tableau D.16 – Codage PACKED_ASCII	693
Tableau D.17 – Codage de BOOLEAN	693

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

BLOCS FONCTIONNELS (FB) POUR LES PROCÉDES INDUSTRIELS ET LE LANGAGE DE DESCRIPTION ÉLECTRONIQUE DE PRODUIT (EDDL) –

Partie 3: Sémantique et syntaxe EDDL

AVANT-PROPOS

- 1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
- 2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
- 3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
- 4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
- 5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
- 6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
- 7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
- 8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
- 9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 61804-3 a été établie par le sous-comité 65E: Les appareils et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels.

Cette troisième édition annule et remplace la deuxième édition parue en 2010. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

- Les Builtins et leurs profils ont été supprimés et déplacés dans la norme IEC 61804-5.
- Les extensions suivantes sont intégrées à la spécification EDDL pour satisfaire aux exigences FDI:

- nouvelles constructions de BLOB, PLUGIN;
 - construction de composant pour les exigences de serveur de communication (attributs supplémentaires);
 - extension de l'attribut classe;
 - nouveaux attributs PRIVATE, VISIBILITY, WRITE_MODE.
- Les modifications suivantes seront intégrées à l'EDDL sur la base de l'harmonisation EDDL:
 - suppression de fonctionnalités non utilisées;
 - harmonisation de fonctionnalités de profil.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65E/451/FDIS	65E/462/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Les titres se terminant par "(vide)" sont utilisés pour conserver la numérotation des éditions précédentes.

Une liste de toutes les parties de la série IEC 61804, publiées sous le titre général *Blocs fonctionnels (FB) pour les procédés industriels et le Langage de description électronique de produit (EDDL)*, peut être consultée sur le site web de l'IEC.

Les futures normes de cette série porteront dorénavant le nouveau titre général cité ci-dessus. Le titre des normes existant déjà dans cette série sera mis à jour lors de la prochaine édition.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "<http://webstore.iec.ch>" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

Le langage EDDL établit un lien entre la spécification conceptuelle de bloc fonction de l'IEC 61804-2 et une implémentation de produit. Il permet aux fabricants d'utiliser la même méthode de description pour des appareils basés sur différentes technologies et diverses plates-formes. La Figure 1 présente ces aspects.

La norme IEC 61804 porte le titre général "Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL)" et comporte les parties suivantes:

Partie 2: Spécification du concept de FB

Partie 3: Sémantique et syntaxe EDDL

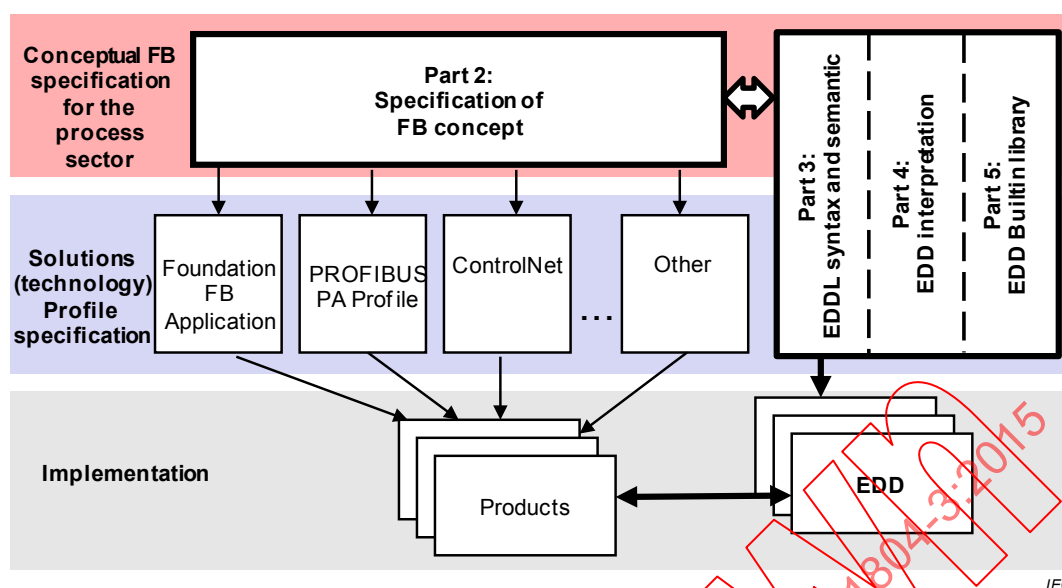
Partie 4: Interprétation EDD

Partie 5: Bibliothèque de Built-in EDDL

Partie 6: Meeting the requirements for integrating fieldbus devices in engineering tools for field devices (disponible en anglais seulement)

Cette partie de la norme IEC 61804 a intégré des parties de la norme IEC TS 61804-1:2003, retirée en janvier 2013.

Le langage EDDL peut également être utilisé pour la description des propriétés de produit pour d'autres domaines tels que l'automatisation industrielle. L'automatisation industrielle peut comprendre des appareils tels que des modules d'entrée/sortie numériques et analogiques génériques, des contrôleurs de mouvement, des interfaces homme-machine, des capteurs, des contrôleurs en boucle fermée, des codeurs, des vannes hydrauliques et des contrôleurs programmables.



IEC

Anglais	Français
Conceptual FB specification for the process sector	Spécification du concept de FB pour le secteur de processus
Part 2: Specification of FB concept	Partie 2: Spécification du concept de FB
Part 3: EDDL syntax and semantic	Partie 3: Sémantique et syntaxe EDDL
Part 4: EDD interpretation	Partie 4: Interprétation EDD
Part 5: EDD Builtin library	Partie 5: Bibliothèque de Builtin EDDL
Solutions (technology) profile specification	Spécification de profils de solutions (technologie)
Foundation FB Application	Application FB Foundation
PROFIBUS PA profile	Profil PROFIBUS PA
ControlNet	ControlNet
Other	Autre
Implementation	Implémentation
Products	Produits
EDD	EDD

Figure 1 – Position de l'IEC 61804 par rapport à d'autres normes et produits

BLOCS FONCTIONNELS (FB) POUR LES PROCÉDES INDUSTRIELS ET LE LANGAGE DE DESCRIPTION ÉLECTRONIQUE DE PRODUIT (EDDL) –

Partie 3: Sémantique et syntaxe EDDL

1 Domaine d'application

La présente partie de la série IEC 61804 spécifie la technologie de langage de description électronique de produit (EDDL¹), qui permet en utilisant les outils d'ingénierie l'intégration des détails du produit réel dans les systèmes tout au long du cycle de vie.

La présente partie de la norme IEC 61804 spécifie l'EDDL en tant que langage générique pour décrire les propriétés des composants système mettant en œuvre des automatismes. L'EDDL est capable de décrire:

- les paramètres des appareils et leurs dépendances;
- les fonctions des appareils, par exemple le mode de simulation, l'étalonnage;
- les représentations graphiques, par exemple les menus;
- les interactions avec les appareils de commande;
- les représentations graphiques:
 - interface utilisateur avancée;
 - système graphique;
- le répertoire des données persistantes.

L'EDDL est utilisé pour créer une description d'appareil électronique (EDD²), par exemple pour les appareils concrets, les profils ou bibliothèques utilisables en commun. Cette EDD est utilisée par des outils appropriés pour générer un code interprété qui prend en charge la manipulation des paramètres, l'exploitation et la surveillance des composants système mettant en œuvre des automatismes, tels que des E/S, des contrôleurs, des capteurs et des contrôleurs programmables à distance. Les outils d'implémentation ne font pas partie du domaine d'application de la présente norme.

La présente partie de l'IEC 61804 spécifie la sémantique et la structure lexicale indépendamment de la syntaxe. Une syntaxe spécifique est définie à l'Annexe A mais le modèle sémantique peut également être utilisé avec d'autres syntaxes.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 60050 (toutes les parties), *Vocabulaire électrotechnique international* (disponible sous <http://www.electropedia.org>)

¹ EDDL = *Electronic Device Description Language*.

² EDD = *Electronic Device Description*.

IEC 61804-2:–3, *Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL) – Partie 2: Spécification du concept de FB et langage de description électronique de produit (EDDL)*

IEC 61804-5, *Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL) – Partie 5: Bibliothèque de Builtin EDDL*

IEC 62541-4, *Architecture unifiée OPC – Partie 4: Services*

ISO/IEC 2375, *Technologies de l'information – Procédure pour l'enregistrement des séquences d'échappement et des jeux de caractères codés*

ISO/IEC 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Le modèle de base*

ISO/IEC 8859-1, *Technologies de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 1: Alphabet latin no. 1*

ISO/IEC 9834-8, *Technologies de l'information – Procédures opérationnelles pour les organismes d'enregistrement d'identificateur d'objet – Partie 8: Génération des identificateurs uniques universels (UUID) et utilisation de ces identificateurs dans les composants d'identificateurs d'objets*

ISO/IEC 9899, *Technologies de l'information – Langages de programmation – C*

ISO/IEC 10646-1, *Technologie de l'information – Jeu universel de caractères codés sur plusieurs octets (JUC) – Partie 1: Architecture et plan multilingue de base*

ISO 639 (toutes les parties), *Codes pour la représentation des noms de langue*

ISO 3166-1, *Codes pour la représentation des noms de pays et de leurs subdivisions – Partie 1: Codes de pays*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic* (disponible en anglais seulement)

RFC 3629, *UTF-8, User Datagram Protocol* (disponible en anglais seulement), disponible à l'adresse <http://www.ietf.org/rfc/rfc0768.txt>

W3C Cascading Style Sheets Level 2 Specification (disponible en anglais seulement) <http://www.w3.org/TR/CSS2>

3 Termes, définitions, termes abrégés et acronymes

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés dans l'IEC 60050-351, dans l'IEC 61804-2 ainsi que les suivants s'appliquent.

3.1.1

builtin

sous-programme exécuté par l'application EDD, pour la communication et l'affichage par exemple.

3.1.2

appareil

entité physique indépendante capable d'accomplir une ou plusieurs fonctions spécifiées dans un contexte particulier et délimitée par ses interfaces

[SOURCE: IEC 61499-1:2012, 3.29]

3.1.3

application EDD

programme utilisant l'EDD, ou une forme traduite quelconque, qui produit une fonctionnalité telle qu'une représentation de communication, une représentation de données, une représentation graphique, etc.

3.1.4

processeur EDDL

processeur ou programme, qui traduit l'EDD en forme exécutable qui peut être traitée par une application EDD

3.1.5

profil EDDL

sélection des éléments pris en charge de la structure lexicale EDDL comprenant les définitions de syntaxe pour une pluralité de consortiums spécifiques

3.1.6

langage de description électronique de produit EDDL

langage formel pour décrire les appareils et les composants de communication

Note 1 à l'article: L'abréviation EDDL est dérivée du terme anglais développé correspondant "Electronic Device Description Language".

3.1.7

description d'appareil électronique EDD

collection de données contenant le(s) paramètre(s) de l'appareil, leur(s) dépendance(s), leur(s) représentation(s) graphique(s) et une description des ensembles de données qui sont transférés.

Note 1 à l'article: La description de l'appareil électronique est créée en utilisant l'EDDL.

Note 2 à l'article: L'abréviation EDD est dérivée du terme anglais développé correspondant "Electronic Device Description".

3.1.8

compilateur de description d'appareil électronique

compilateur qui traduit la source d'EDD dans un format interne qui est utilisé par l'interpréteur d'EDD

3.1.9

interpréteur de description d'appareil électronique EDDI

interpréteur qui utilise la source d'EDD ou un format interne qui est donné par le compilateur EDDL pour fournir les informations d'EDD à l'utilisateur d'EDD

Note 1 à l'article: L'abréviation EDDI est dérivée du terme anglais développé correspondant "Electronic Device Description Interpreter".

3.1.10

fonction bloc

instance de fonction bloc

unité fonctionnelle logicielle comprenant une copie individuelle nommée d'une structure de données et des opérations associées spécifiées par un type de FB correspondant

Note 1 à l'article: Des opérations typiques d'un FB comprennent une modification des valeurs des données dans sa structure de données associée.

[SOURCE: IEC 61804-2:–, 3.1.32]

3.1.11

matériel

équipement physique, par opposition à des programmes, des procédures, des règles et documentation associée

3.1.12

préprocesseur

partie de l'environnement d'exécution qui transforme les informations fournies dans une EDD

3.1.13

directives de préprocesseur

description de conditions pour filtrer le code d'EDD avant compilation ou interprétation

Note 1 à l'article: Par exemple, une directive de préprocesseur permettant de définir des noms pour des constantes ou décrire des macros pour rendre le code plus facile à lire.

3.1.14

logiciel

création intellectuelle comprenant les programmes, procédures, règles et documentation associée relatifs au fonctionnement d'un système

[SOURCE: IEC 61499-1:2012, 3.92]

3.2 Termes abrégés et acronymes

ASCII	Code normalisé américain pour l'échange d'informations (American Standard Code for Information Interchange) conformément à l'ISO/IEC 10646-1
CP	Profil de communication (Communication Profile)
CPF	Famille de profils de communication (Communication Profile Family)
EDD	Description d'appareil électronique (Electronic Device Description)
EDDL	Langage de description électronique de produit (Electronic Device Description Language)
EUC	Unicode étendu (Extended Unicode) conformément à l'ISO/IEC 2022
FB	Fonction bloc (Function block)
FF	Fieldbus Foundation ⁴
HCF	HART® ⁵ Communication Foundation
HMI	Interface homme machine (Human-machine interface)
E/S	Entrée/Sortie (Input/Output)
ID	Identificateur (Identifier)
OSI	Interconnexion de systèmes ouverts (Open Systems Interconnection)

⁴ FOUNDATION™ Fieldbus est l'appellation commerciale du consortium Fieldbus Foundation (organisation à but non lucratif). Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande le détenteur de l'appellation commerciale ou l'un quelconque de ses produits. La conformité à ce profil n'exige pas l'utilisation de l'appellation commerciale. L'utilisation des appellations commerciales exige l'autorisation du propriétaire du nom commercial.

⁵ HART® est une marque déposée du consortium HART Communication Foundation (HCF) (organisation à but non lucratif). Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande le détenteur de l'appellation commerciale ou l'un quelconque de ses produits. La conformité à ce profil n'exige pas l'utilisation de l'appellation commerciale. L'utilisation des appellations commerciales exige l'autorisation du propriétaire du nom commercial.

PI	PROFIBUS and PROFINET ⁶ International
PDU	Unité de données de protocole (Protocol data unit)
PNO	PROFIBUS Nutzerorganisation

4 Déclaration de conformité

Une déclaration de conformité à un profil de la présente partie de l'IEC 61804 doit être établie⁷ comme étant

– conforme à l'IEC 61804-3:–, Annexe D.n <Type>

où le Type entre les crochets obliques < > est facultatif et les crochets obliques ne doivent pas être inclus.

Le Type peut par exemple être le nom du protocole de communication utilisé, tel que PROFIBUS, Foundation Fieldbus ou HART Communication Foundation, mais aussi être codé conformément aux profils de communication spécifiés dans la norme IEC 61784-1 ou dans la norme IEC 61784-2.

Les normes de produit ne doivent pas nécessairement comprendre des aspects d'évaluation de conformité (comprenant des dispositions QM), normatifs ou informatifs, autres que des dispositions pour la soumission à l'essai de produit (évaluation et examen).

5 Conventions

5.1 Généralités

Dans la série IEC 61804, tous les mots clés sont écrits en majuscules. Un élément EDDL est écrit en minuscules pour traiter la sémantique de l'ensemble de l'élément.

Le langage EDDL est généralement décrit à l'aide de structures lexicales dans lesquelles les éléments et la présence de champs sont spécifiés.

5.2 Conventions pour la structure lexicale

5.2.1 ABC champ1, champ2

ABC est un élément lexical. Cet élément doit être codé dans une syntaxe concrète. Cet élément ne doit pas forcément être codé avec le nom "ABC". Cet élément peut également être codé, par exemple, avec un numéro de balise (Tag).

Champ1 et champ2 sont des champs de l'élément lexical ABC. Chaque champ est obligatoire et peut avoir plus d'un attribut. Si un champ a des attributs, la présence des attributs est spécifiée dans un tableau. Une virgule sépare champ1 et champ2. La virgule est un élément lexical et n'est pas codée de manière explicite.

⁶ PROFIBUS et PROFINET sont des appellations commerciales de PROFIBUS Nutzerorganisation e.V. (PNO), filiale de PROFIBUS and PROFINET International. PNO est une organisation à but non lucratif pour la prise en charge des technologies de bus de terrain PROFIBUS et PROFINET. Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande le détenteur de l'appellation commerciale ou l'un quelconque de ses produits. La conformité à ce profil n'exige pas l'utilisation de l'appellation commerciale. L'utilisation des appellations commerciales exige l'autorisation du propriétaire du nom commercial.

⁷ Conformément aux directives ISO/IEC.

Si un champ a des attributs additionnels, les attributs sont définis dans un tableau. La disposition du tableau et les qualificatifs d'utilisation possibles sont présentés dans le Tableau 1.

Tableau 1 – Descriptions d'attributs de champ

Utilisa-tion	Attribut	Description
m	yyy	La présence de cet attribut est obligatoire.
o	xxx	La présence de cet attribut est facultative.
s	z1	La présence de cet attribut est sélectionnable avec d'autres attributs, qui sont également marqués avec "s" dans la colonne Utilisation. Un seul des attributs sélectionnables (z1 ou z2) est présent.
s	z2	La présence de cet attribut est sélectionnable avec d'autres attributs, qui sont également marqués avec "s" dans la colonne Utilisation. Un seul des attributs sélectionnables (z1 ou z2) est présent.
c	uuu	La présence de cet attribut est conditionnelle et celui-ci est présent uniquement si la condition est vraie.

Les caractères dans la colonne Utilisation ont les significations suivantes:

m: cet attribut est obligatoire et doit être présent.

o: cet attribut est facultatif et peut ne pas être présent.

s: cet attribut est une sélection. Un et un seul des champs marqués avec "s" (z1 ou z2) doit être présent.

c: cet attribut est conditionnel. La condition est décrite dans la colonne Description.

Dans la colonne Attribut, s'il existe plusieurs attributs qui ont la même utilisation, ceux-ci sont triés par ordre alphabétique.

5.2.2 ABC champ1+

Le signe plus (+) après champ1 est utilisé pour indiquer que champ1 est utilisé au moins une fois. Celui-ci peut être utilisé plus d'une fois.

5.2.3 ABC champ2*

L'astérisque (*) derrière champ2 est utilisé pour indiquer que champ2 est facultatif et peut ne pas être utilisé. S'il est utilisé, il peut être utilisé plus d'une fois.

5.2.4 ABC [champ1, champ2]+

Les éléments champ1 et champ2 entre les crochets [] sont une liste non triée. Le signe plus (+) derrière le crochet final indique que champ1 et champ2 sont utilisés au moins une fois et peuvent être utilisés plus d'une fois en tant que groupe.

5.2.5 ABC champ1, (champ2, champ3)<exp>

<exp> indique que champ2 et champ3 sont utilisés conjointement avec l'expression conditionnelle (des constructions conditionnelles sont spécifiées en 7.37). L'expression <exp> désigne uniquement les champs entre parenthèses (). L'utilisation d'expressions conditionnelles est facultative.

6 Modèle EDD et EDDL

6.1 Présentation des EDD et du langage EDDL

Une description d'appareil électronique (EDD) contient tous les paramètres d'un appareil d'un composant de système d'automatismes. La technologie utilisée pour décrire une EDD est appelée langage de description électronique de produit (EDDL). Le langage EDDL comporte un ensemble d'éléments de langage basiques évolutifs pour gérer des appareils simples, complexes ou modulaires. Le langage EDDL est un langage descriptif basé sur un format ASCII avec une séparation nette entre les données et le programme.

Les données dans un champ de texte, qui sont marquées avec un code de pays (par exemple, japonais) peuvent utiliser un code multi-octets.

6.2 Architecture EDD

Du point de vue du modèle ISO/OSI (ISO/IEC 7498-1), une EDD est au-dessus de la couche 7. L'application EDD utilise toutefois son système de communication pour transférer ses informations. Une EDD contient des constructions qui prennent en charge le mapping vers un système de communication de prise en charge.

Le fabricant de l'appareil définit les objets, qui formeront la représentation logique des objets d'une application EDD. Pour cette raison, EDDL comprend des éléments de langage, qui mettent en correspondance les données de l'EDD avec la représentation de données du système de communication, de sorte que l'utilisateur d'une EDD n'a pas besoin de connaître l'emplacement physique (adresse) de l'objet appareil.

Une EDD décrit la gestion des informations à afficher à l'utilisateur. La représentation spécifique d'une telle visualisation ne fait pas partie des définitions d'EDD ou EDDL.

6.3 Concepts d'EDD

Le fabricant d'un appareil ou d'un composant d'un système d'automatismes décrit les propriétés de l'appareil en utilisant l'EDDL. L'EDD résultant contient des informations telles que:

- la description des paramètres de l'appareil;
- la description des dépendances des paramètres;
- le regroupement logique des paramètres de l'appareil;
- la sélection et l'exécution des fonctions de l'appareil prises en charge;
- la description des ensembles de données transférés.

Suivant l'utilisation exigée, l'EDD peut être physiquement située:

- dans un appareil;
- dans un support de stockage de données externe tel qu'un disque compact, une disquette ou un serveur;
- à la fois dans l'appareil et dans un support de stockage externe.

Une EDD prend en charge les chaînes de texte (termes communs, phrases, etc.) en plusieurs langues (anglais, allemand, français, etc.). Les chaînes de texte peuvent être stockées dans des dictionnaires séparés. Il peut y avoir plus d'un dictionnaire pour une EDD.

Une implémentation d'EDD comprend des informations nécessaires concernant l'appareil cible, par exemple, le fabricant, le type d'appareil, la révision, etc. Elles sont utilisées pour mettre en correspondance une EDD spécifique avec un appareil spécifique.

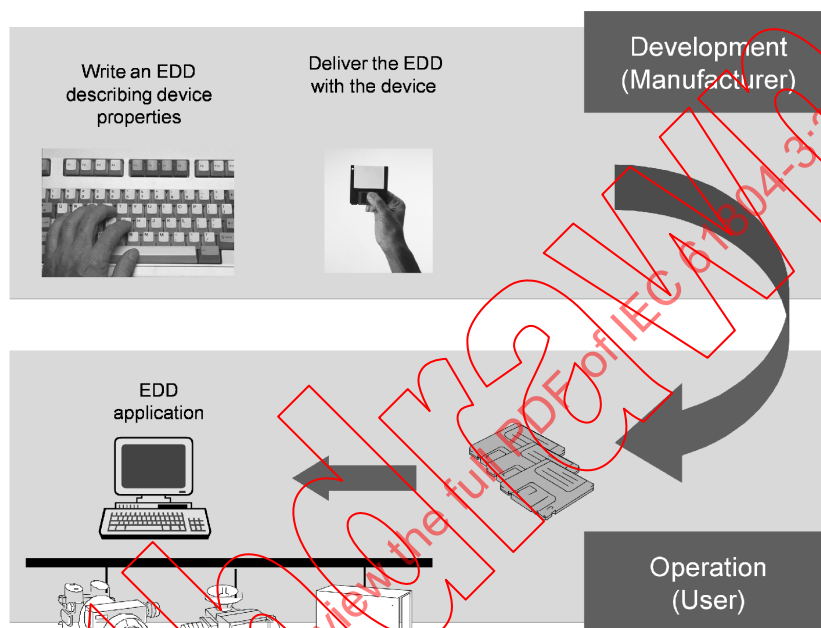
6.4 Principes du processus de développement d'EDD

6.4.1 Généralités

Le processus de création d'une EDD comporte trois étapes: génération du code source de l'EDD, prétraitement de l'EDD et compilation de l'EDD.

6.4.2 Génération du code source de l'EDD

Le processus de génération d'EDD est représenté sur la Figure 2.



IEC

Anglais	Français
Write an EDD describing device properties	Ecriture d'une EDD décrivant les propriétés de l'appareil
Deliver the EDD with the device	Fourniture de l'EDD avec l'appareil
Development (manufacturer)	Développement (fabricant)
EDD application	Application EDD
Operation (user)	Opération (utilisateur)

Figure 2 – Processus de génération d'EDD

Le fabricant d'un appareil écrit une EDD appropriée pour son appareil et fournit les deux (l'EDD et l'appareil). Si le système mettant en œuvre des automatismes prend en charge la méthode EDD, une intégration de système peut être effectuée par l'utilisateur.

Les EDD pour les appareils peuvent être imbriquées dans la mémoire de l'appareil ou fournies à l'aide de supports de stockage séparés ou téléchargées depuis un serveur réseau approprié. L'EDD est "interprétée" ou "parcourue" par une application EDD. Les EDD sont normalement stockées en tant que fichiers sources ou fichiers prétraités.

6.4.3 Prétraitement de l'EDD

Dans l'étape de prétraitement, un préprocesseur d'EDD génère une représentation de l'EDD cohérente et adaptée pour une compilation finale.

Le prétraitement prend en charge, par exemple, la substitution des définitions et l'inclusion de texte externe. La sortie du préprocesseur est une EDD complète sans aucune directive de préprocesseur.

6.4.4 Compilation de l'EDD

L'étape de compilation d'EDD produit une représentation interne d'application EDD à partir d'une EDD prétraitée destinée à être utilisée dans l'application EDD.

6.5 Interrelations entre la structure lexicale et les définitions formelles

La structure lexicale de l'EDDL et ses éléments sont décrits à l'Article 7. La syntaxe et les définitions formelles de chaque élément EDDL sont données à l'Annexe A. La structure lexicale et sa description formelle utilisent le même nom.

NOTE A la place des définitions et de la syntaxe formelles spécifiées dans l'Annexe A, une autre spécification peut être développée en tant qu'option additionnelle.

6.6 Builtins

Les Builtins sont des sous-programmes prédéfinis qui sont exécutés dans l'application EDD.

EXEMPLE Un terminal portatif est un appareil simple ayant un petit écran et des fonctions de curseur limitées. Pour ce type d'appareil, un Builtin pourrait être spécifié pour produire une entrée d'affichage en utilisant uniquement des actions de curseur haut/bas gauche/droite.

La bibliothèque de Builtins est définie dans la norme IEC 61804-5.

6.7 Profils

L'EDDL est une spécification harmonisée de concepts EDD existants. Les applications EDD concrètes utilisent un sous-ensemble de la spécification EDDL. La sélection d'éléments EDDL et de Builtins est effectuée à partir des profils définis dans l'Annexe D.

En plus des profils de l'EDD, les consortiums implémenteurs publient également des "profils d'appareil", qui sont utilisés pour prendre en charge l'interchangeabilité d'appareils conformes. Ces profils d'appareils peuvent être décrits en utilisant l'EDDL spécifié dans le présent document.

7 Langage de description électronique de produit (EDDL)

7.1 Vue d'ensemble

7.1.1 Caractéristiques du langage EDDL

L'EDDL est un langage structuré et interprétatif pour décrire les propriétés d'un appareil. De plus, les interactions entre les appareils et l'environnement d'exécution de l'EDD sont incorporés dans l'EDDL. Le langage EDDL comprend un ensemble d'éléments de langage pour ces applications.

Pour une implémentation EDD spécifique, tous les éléments fournis par le langage ne doivent pas être utilisés.

Des sous-ensembles compatibles du langage EDDL sont admis et peuvent être spécifiés en utilisant des profils (par exemple, le choix de constructions, le nombre de récurrences et la sélection d'options). Les profils pris en charge par certains consortiums industriels sont spécifiés dans l'Annexe D. Les développeurs d'EDD doivent identifier dans chaque appareil les détails sur le profil qui a été utilisé.

7.1.2 Représentation de syntaxe

La spécification du langage EDDL dans l'Article 7 de la présente partie de l'IEC 61804 utilise une syntaxe abstraite. La structure lexicale abstraite du 7.1 est convertie en syntaxes spécifiques spécifiées dans l'Annexe A par le nom de l'élément (par exemple, VARIABLE dans la structure lexicale équivaut au mot-clé VARIABLE dans l'Annexe A).

Hormis la syntaxe définie dans la présente partie de l'IEC 61804, d'autres définitions de syntaxe peuvent être utilisées et peuvent être ajoutées dans le futur pour permettre d'autres caractéristiques et représentations pour une évolution future.

7.1.3 Éléments de langage d'une EDD

Le langage est structuré avec les éléments de langage suivants:

- éléments d'identification;
- éléments de construction de base;
- éléments spécifiques.

Il convient que les chaînes de texte fréquemment utilisées (un texte d'aide et des listes multilingues d'étiquettes, par exemple) soient séparées dans un dictionnaire de texte (voir 7.41).

Les informations d'identification doivent constituer la première entrée dans chaque fichier EDD et doivent apparaître une seule fois. Les informations d'identification identifient uniquement la version du langage EDDL utilisée, conjointement avec le type d'appareil, les codes du modèle et les détails de révision spécifiques couverts par le fichier EDD.

7.1.4 Éléments de construction de base

7.1.4.1 Généralités

Ces constructions de base ont été spécifiées pour prendre en charge les descriptions d'appareils utilisés dans les applications de contrôle industriel, conjointement avec leurs propriétés et fonctionnalités.

Certaines constructions ont des noms et fonctions similaires mais diffèrent dans leurs spécifications détaillées. Cette variété additionnelle a été incluse afin d'assurer la compatibilité avec plusieurs langages de description existants. Des références croisées de mapping appropriées sont données par les profils.

Chacune des constructions de base a un jeu d'attributs associé. Les attributs peuvent avoir des sous-attributs, qui affinent la définition de l'attribut et par conséquent, la définition de la construction elle-même.

La définition d'un attribut peut être statique ou dynamique. Une définition d'attribut statique ne change jamais, tandis qu'une définition d'attribut dynamique peut changer afin de prendre en compte des changements de valeur de paramètres.

7.1.4.2 AXIS

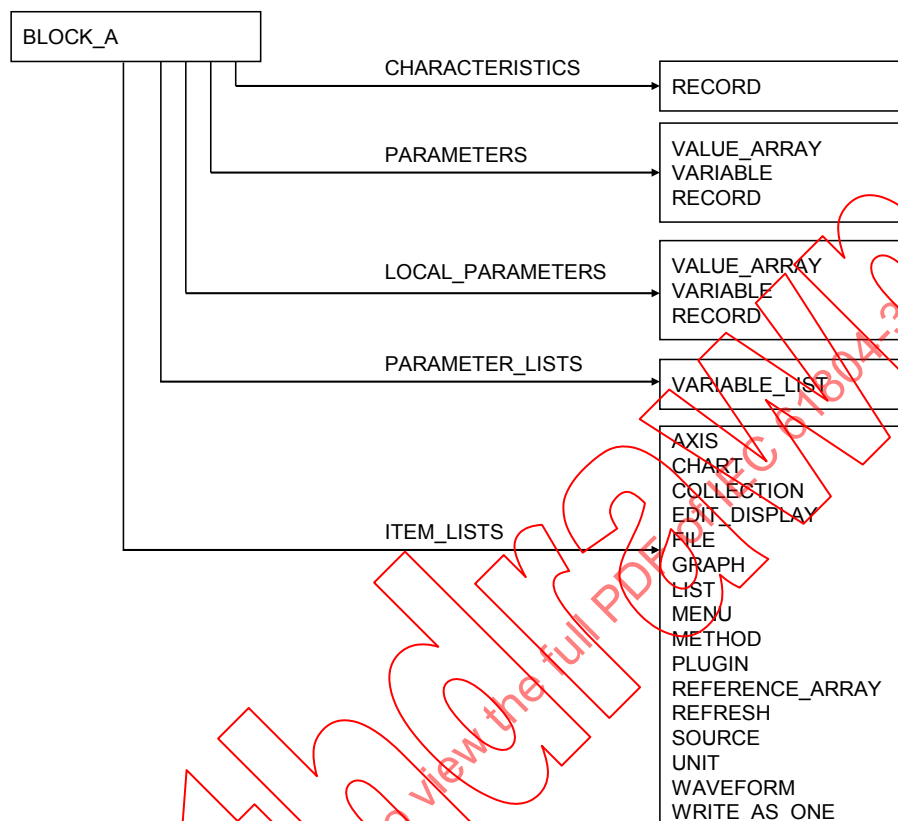
AXIS décrit l'axe d'un attribut CHART ou GRAPH (voir 7.3).

7.1.4.3 BLOB

BLOB, Binary Large Object, est un objet binaire grand utilisé pour transférer les données binaires depuis ou vers un appareil (voir 7.43).

7.1.4.4 BLOCK_A

BLOCK_A est un regroupement logique d'attributs CHARACTERISTICS, PARAMETERS, PARAMETER_LISTS et ITEM_LISTS; voir Figure 3. Pour accéder à un élément de BLOCK_A, il convient d'utiliser l'instance du bloc (voir 7.4.1 et Figure 3).



IEC

Figure 3 – BLOCK_A

7.1.4.5 BLOCK_B

BLOCK_B est utilisé pour structurer les variables en instances de types de blocs logiques. Pour accéder à un attribut VARIABLE dans une construction BLOCK_B, la construction de base COMMAND est utilisée pour générer l'adressage relatif (voir 7.4.2).

7.1.4.6 CHART

CHART décrit un diagramme utilisé pour afficher des données provenant d'un appareil dans une application EDD (voir 7.5 et Figure 4).



IEC

Figure 4 – CHART

7.1.4.7 COLLECTION

COLLECTION décrit des regroupements logiques de données (voir 7.6 et Figure 5).

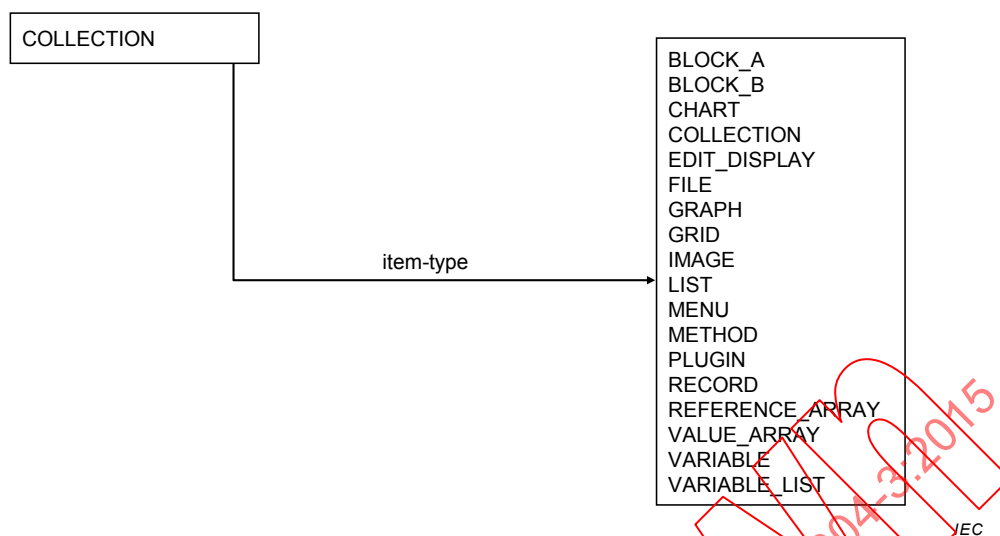


Figure 5 – COLLECTION

7.1.4.8 COMMAND

COMMAND décrit la structure et l'adressage des variables dans l'appareil (voir 7.7 et Figure 6).

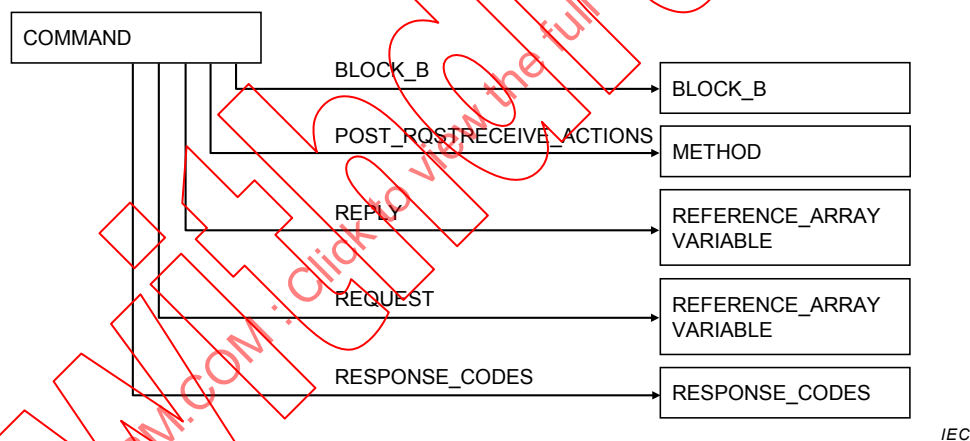


Figure 6 – COMMAND

7.1.4.9 COMPONENT

COMPONENT décrit la configuration du comportement d'un appareil qui est décrit dans l'EDD (voir 7.8 et Figure 7).

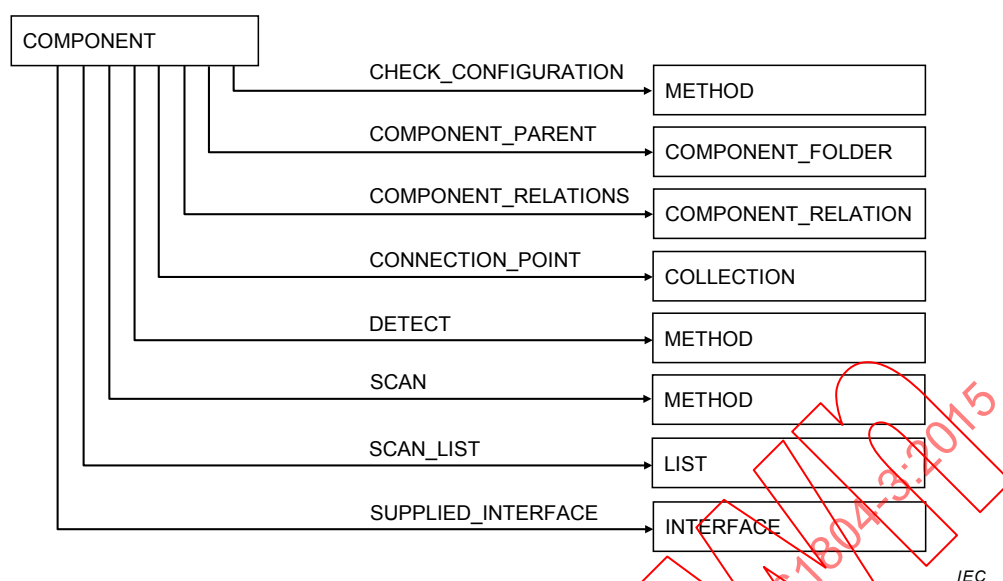


Figure 7 – COMPONENT

7.1.4.10 COMPONENT_FOLDER

COMPONENT_FOLDER décrit un dossier dans un catalogue. Les dossiers permettent de regrouper des composants dans un catalogue (voir 7.9 et Figure 8).



Figure 8 – COMPONENT_FOLDER

7.1.4.11 COMPONENT_REFERENCE

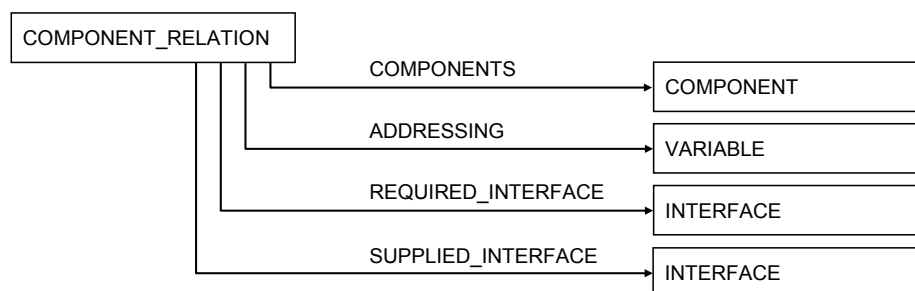
COMPONENT_REFERENCE fait référence à un attribut **COMPONENT** ou **COMPONENT_FOLDER** (voir 7.10 et Figure 9).



Figure 9 – COMPONENT_REFERENCE

7.1.4.12 COMPONENT_RELATION

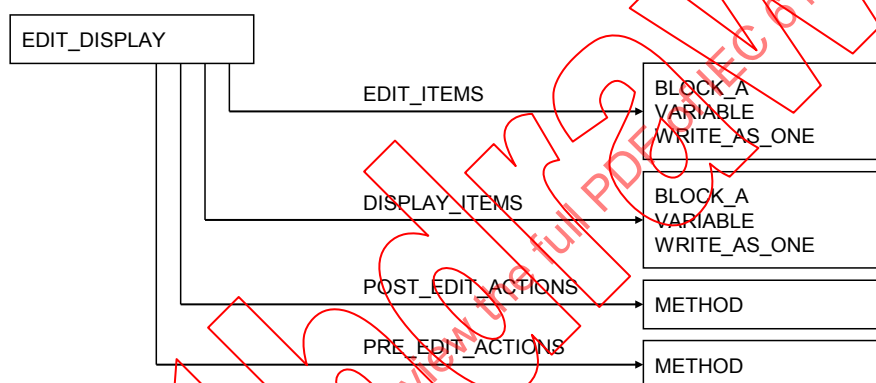
COMPONENT_RELATION spécifie une relation entre des attributs **COMPONENT**, par exemple un module et un ou plusieurs autres modules (voir 7.11 et Figure 10).



IEC

Figure 10 – COMPONENT_RELATION**7.1.4.13 EDIT_DISPLAY**

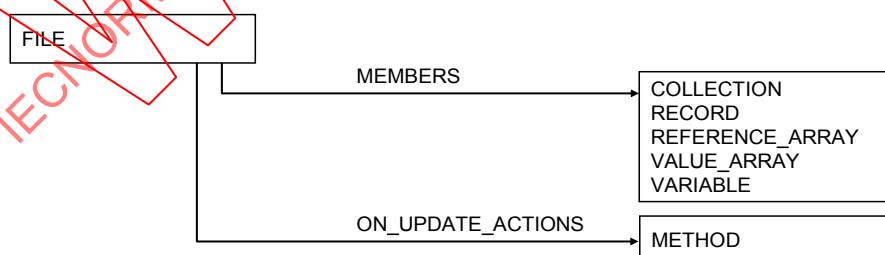
EDIT_DISPLAY décrit la manière dont les données seront présentées à un utilisateur par un appareil d'affichage (voir 7.14 et Figure 11).



IEC

Figure 11 – EDIT_DISPLAY**7.1.4.14 FILE**

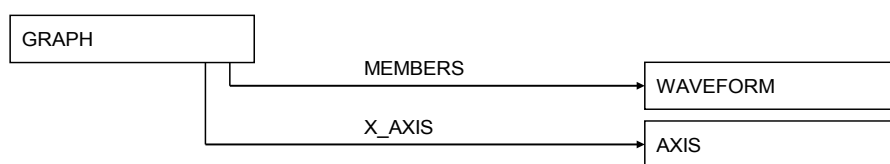
FILE décrit une zone de stockage persistant (voir 7.15 et Figure 12).



IEC

Figure 12 – FILE**7.1.4.15 GRAPH**

GRAPH décrit un graphique utilisé pour afficher des données provenant d'un appareil dans l'application EDD (voir 7.16 et Figure 13).



IEC

Figure 13 – GRAPH

7.1.4.16 GRID

GRID décrit une matrice utilisée pour afficher des données provenant d'un appareil dans l'application EDD (voir 7.17 et Figure 14).

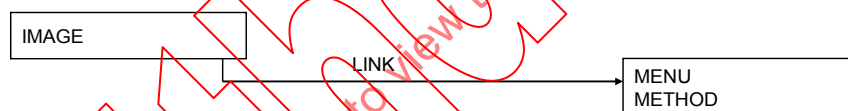


IEC

Figure 14 – GRID

7.1.4.17 IMAGE

IMAGE décrit une image visuelle (voir 7.18 et Figure 15).



IEC

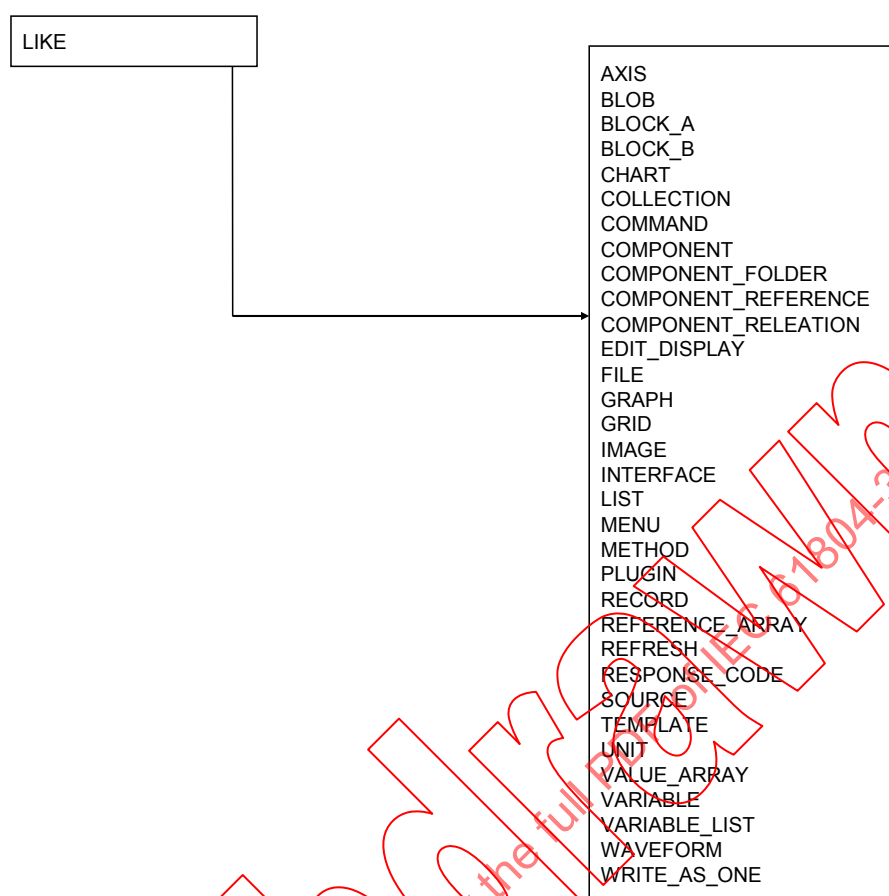
Figure 15 – IMAGE

7.1.4.18 IMPORT

IMPORT est utilisé pour importer une EDD et modifier ses définitions existantes (voir 7.19).

7.1.4.19 LIKE

LIKE crée une nouvelle instance d'une instance existante d'une construction de base. Les attributs de la nouvelle instance peuvent être redéfinis, ajoutés ou supprimés (voir 7.21 et Figure 16).

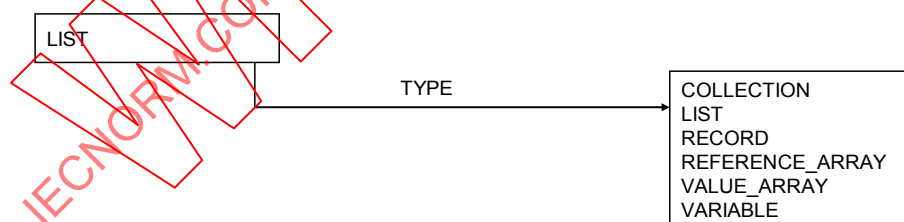


IEC

Figure 16 – LIKE

7.1.4.20 LIST

LIST décrit une séquence d'éléments (voir 7.22 et Figure 17).



IEC

Figure 17 – LIST

7.1.4.21 MENU

MENU décrit la manière dont les données seront présentées à un utilisateur par une application EDD (voir 7.23 et Figure 18).

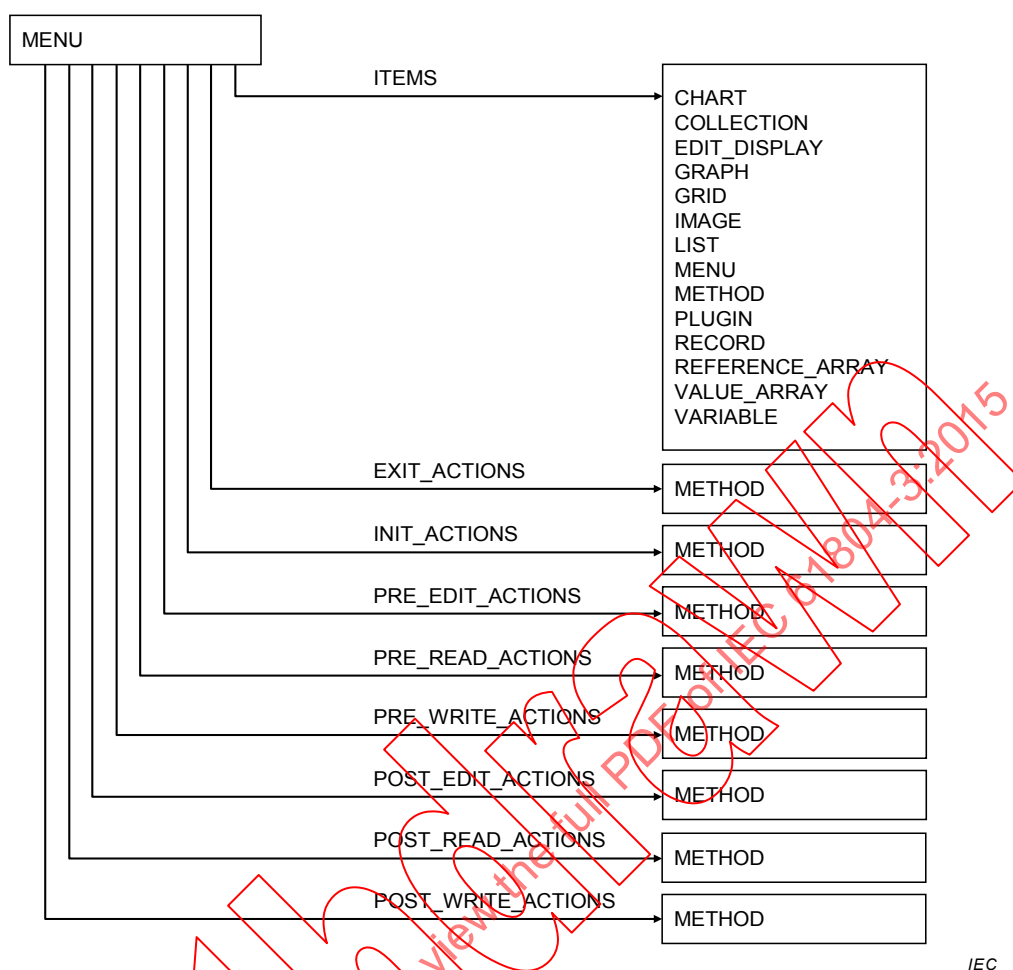


Figure 18 – MENU

7.1.4.22 METHOD

METHOD décrit l'exécution d'une séquence complexe d'interactions d'événements qui doit se produire entre les appareils du système tels que des unités d'affichage, des configurateurs et des appareils (voir 7.24).

7.1.4.23 PLUGIN

PLUGIN décrit un plug-in d'interface utilisateur (UIP, User Interface Plug-in) tel que décrit dans 7.42.

7.1.4.24 RECORD

RECORD décrit les données contenues dans l'appareil (voir 7.26 et Figure 19).

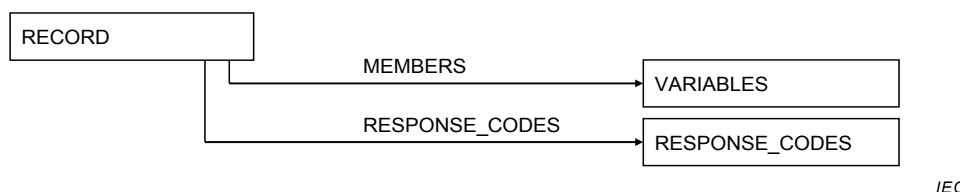
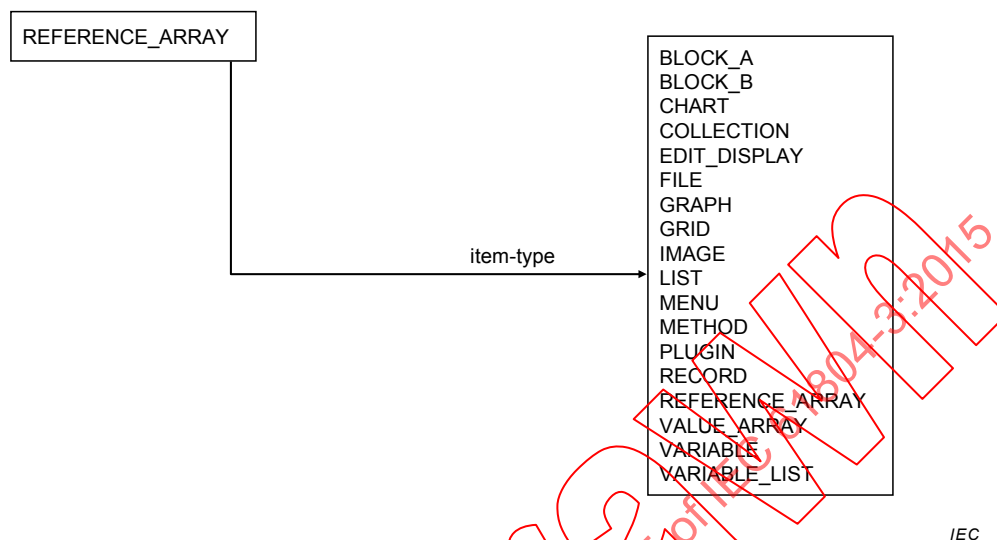


Figure 19 – RECORD

7.1.4.25 REFERENCE_ARRAY

REFERENCE_ARRAY est un ensemble d'éléments EDD du même type d'élément EDDL (par exemple, VARIABLE ou MENU). Un élément peut être référencé dans l'EDD via le nom de REFERENCE_ARRAY et l'index associé à l'élément (voir 7.27 et Figure 20).



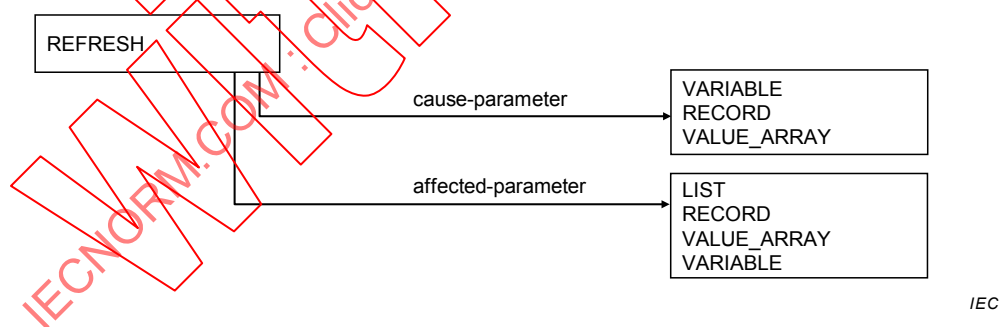
IEC

Figure 20 – REFERENCE_ARRAY

7.1.4.26 Relations

7.1.4.26.1 REFRESH

REFRESH décrit les relations entre les attributs LIST, RECORD, VALUE_ARRAY et VARIABLE (voir 7.28.1 et Figure 21).



IEC

Figure 21 – REFRESH

7.1.4.26.2 UNIT

UNIT décrit les relations entre les attributs AXIS, VALUE_ARRAY et VARIABLE (voir 7.28.2 et Figure 22).

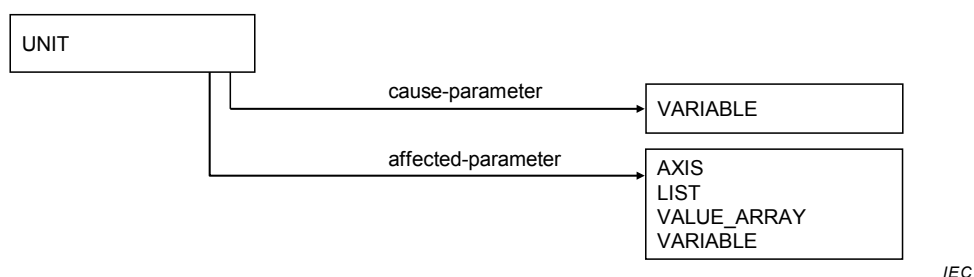


Figure 22 – UNIT

7.1.4.26.3 WRITE_AS_ONE

WRITE_AS_ONE décrit les relations entre les attributs VARIABLE, RECORD et VALUE_ARRAY (voir 7.28.3 et Figure 23).



Figure 23 – WRITE_AS_ONE

7.1.4.27 RESPONSE_CODES

RESPONSE_CODES spécifie la liste des éléments de code de réponse qui doivent être utilisés par l'application EDD pour informer l'utilisateur du résultat de la communication, par exemple erreur, avertissement (voir 7.29).

7.1.4.28 SOURCE

SOURCE décrit une source de valeurs de données pour un attribut CHART (voir 7.30 et Figure 24).

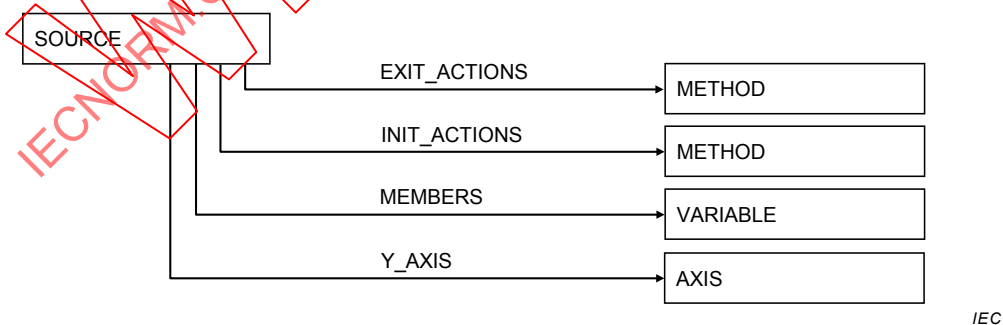
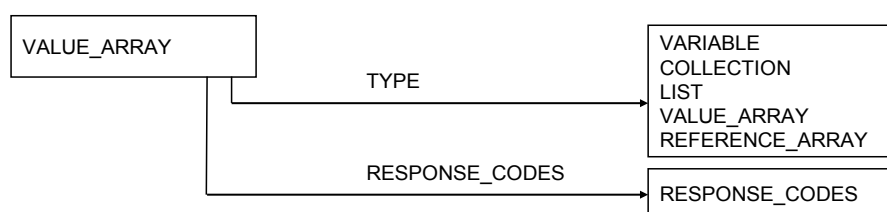


Figure 24 – SOURCE

7.1.4.29 VALUE_ARRAY

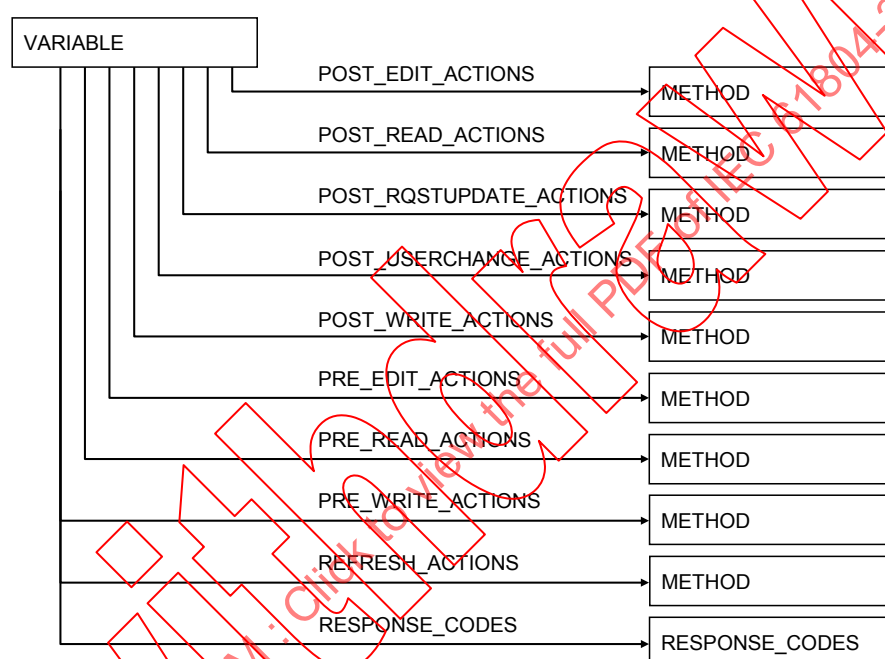
VALUE_ARRAY décrit des groupes logiques de variables EDDL (voir 7.32 et Figure 25).



IEC

Figure 25 – VALUE_ARRAY**7.1.4.30 VARIABLE**

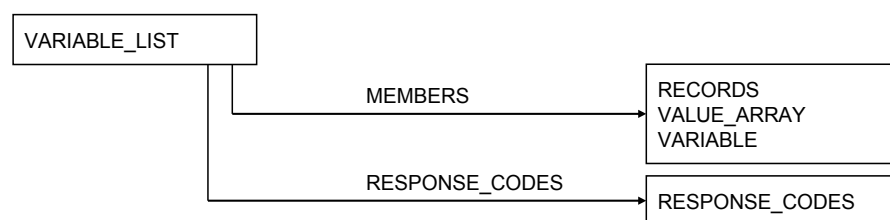
VARIABLE décrit les données contenues dans l'appareil (voir 7.33 et la Figure 26).



IEC

Figure 26 – VARIABLE**7.1.4.31 VARIABLE_LIST**

VARIABLE_LIST décrit les regroupements logiques de données contenues dans l'appareil qui peuvent être communiqués sous forme de liste (voir 7.34 et Figure 27).



IEC

Figure 27 – VARIABLE_LIST

7.1.4.32 WAVEFORM

WAVEFORM décrit un ensemble de données qui peut être affiché par un attribut GRAPH (voir 7.35 et Figure 28).

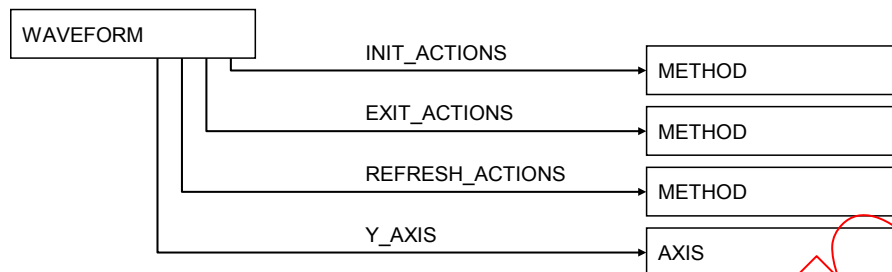


Figure 28 – WAVEFORM

7.1.5 Attributs communs

Les attributs communs suivants sont utilisés dans de nombreuses constructions de base:

- CLASSIFICATION
- COMPONENT_PARENT
- COMPONENT_PATH
- DEFINITION
- EMPHASIS
- HANDLING
- HEIGHT
- HELP
- IDENTITY
- LABEL
- LINE_COLOR
- LINE_TYPE
- MEMBERS
- PRIVATE
- PROTOCOL
- RESPONSE_CODES
- SUPPLIED_INTERFACE
- VALIDITY
- VISIBILITY
- WIDTH
- WRITE_MODE

7.1.6 Éléments spéciaux

Les éléments spéciaux sont des mécanismes EDDL additionnels utilisés pour prendre en charge la gestion des fichiers, les instances multiples et la modification d'éléments de base EDDL, tels que:

- les **expressions conditionnelles** qui spécifient des valeurs d'attributs, qui sont dépendantes des valeurs de variables à l'exécution;
- une **référence** qui est utilisée dans l'ensemble d'une description d'appareil pour désigner d'autres éléments dans le même EDDL;
- une **expression** qui est une opération logique ou mathématique sur des valeurs de variable afin de modifier des valeurs d'attribut d'une construction de base à l'exécution.

7.1.7 Règles pour les instances

Chaque instance d'une construction de base doit être identifiée par un identificateur de type chaîne de caractères. Cet identificateur doit être unique dans l'ensemble de l'EDD. Toutes les constructions de l'EDDL représentent des instances réelles d'informations de l'appareil. Si un appareil contient deux éléments d'information ou plus qui ont la même structure, la construction appropriée doit être répétée dans l'EDD avec des identificateurs différents. En variante, la construction de LIKE peut être utilisée pour créer une copie de la construction avec un identificateur différent. L'identificateur dans la partie de définition lexicale n'est pas décrit à nouveau dans chaque construction.

7.1.8 Règles pour une liste d'attributs VARIABLE

Une liste d'attributs VARIABLE peut contenir des VARIABLE, des éléments de types composés ou des types composés. Les types composés sont:

- BLOCK_A PARAMETERS
- COLLECTIONS
- RECORDS
- REFERENCE_ARRAY
- VALUE_ARRAY
- VARIABLE_LISTS

Au lieu de référencer chaque membre individuel d'un type composé, l'instance de type peut être référencée.

7.2 Informations d'identification d'une EDD

7.2.1 Structure générale

Objet

Les informations d'identification d'une EDD identifient de façon unique la description de l'appareil d'un type d'appareil spécifique d'un fabricant d'appareils. Elles spécifient également la version de l'EDDL et le profil EDDL utilisé.

Ces informations d'identification doivent constituer la première entrée dans chaque EDD et doivent apparaître une seule fois. L'ordre strict des mots clés des informations d'identification d'une EDD n'est pas impératif.

Structure lexicale

DD_REVISION, DEVICE_REVISION, DEVICE_TYPE, EDD_PROFILE, EDD_VERSION,
MANUFACTURER, MANUFACTURER_EXT

7.2.2 Attributs spécifiques

7.2.2.1 DD_REVISION

Objet

L'attribut DD_REVISION spécifie un code unique défini par le fabricant pour la révision d'une EDD de l'appareil.

Il convient que le fabricant modifie le numéro de révision chaque fois qu'une nouvelle description d'appareil est émise pour un appareil.

Il peut exister plusieurs EDD pour une révision particulière d'un appareil particulier, auquel cas l'attribut DD_REVISION permet de les distinguer.

Structure lexicale

DD_REVISION integer

L'attribut DD_REVISION est spécifié dans le Tableau 2.

Tableau 2 – Attribut DD_REVISION

Utilisation	Attribut	Description
m	integer	Entier non signé à deux octets contenant le code pour une révision de fichier EDD spécifique.

7.2.2.2 DEVICE_REVISION

Objet

L'attribut DEVICE_REVISION spécifie un code unique pour la révision de l'appareil, il est défini par le fabricant.

Structure lexicale

DEVICE_REVISION integer

L'attribut DEVICE_REVISION est spécifié dans le Tableau 3.

Tableau 3 – Attribut DEVICE_REVISION

Utilisation	Attribut	Description
m	integer	Entier non signé à deux octets contenant le code pour une révision de l'appareil spécifique.

7.2.2.3 DEVICE_TYPE

Objet

L'attribut DEVICE_TYPE spécifie un identificateur pour le type d'appareil, il est défini par le fabricant de l'appareil. Il convient que l'identificateur soit unique pour chaque type.

Le type d'appareil peut spécifier une catégorie d'appareils ou un produit spécifique.

Structure lexicale

DEVICE_TYPE (integer, identifier)

Les attributs DEVICE_TYPE sont spécifiés dans le Tableau 4.

Tableau 4 – Attributs DEVICE_TYPE

Utilisa-tion	Attribut	Description
s	integer	Entier non signé à deux octets contenant le code pour un type d'appareil ou groupe de types d'appareil spécifique.
s	identifier	Référence à un entier à deux octets qui contient le code pour un type d'appareil ou groupe de types d'appareil spécifique.
Le mapping de l'identificateur à une constante entière peut être stocké dans un fichier séparé appelé "devices.cfg". Ce fichier contient une liste d'identificateurs pour des appareils ou groupes de types d'appareil.		

7.2.2.4 EDD_PROFILE

Objet

L'attribut EDD_PROFILE spécifie le profil sélectionné d'EDDL, il est défini dans l'Annexe D.

Structure lexicale

EDD_PROFILE integer

L'attribut EDD_PROFILE est spécifié dans le Tableau 5.

Tableau 5 – Attribut EDD_PROFILE

Utilisa-tion	Attribut	Description
o	integer	Entier non signé à deux octets contenant le code pour les profils spécifiques sélectionnés dans l'Annexe D. Les codes suivants sont actuellement définis: 0x01 Profil pour HART Communication Foundation (voir Article D.4) 0x02 Profil pour Fieldbus Foundation (voir Article D.3) 0x03 Profil pour PROFIBUS (voir Article D.2) 0x04 Profil pour PROFINET (voir Article D.2)

7.2.2.5 EDD_VERSION

Objet

L'attribut EDD_VERSION spécifie un code unique pour la version d'EDD, qui a été utilisé par le fabricant dans la construction de l'EDD. Cet attribut doit être inclus avec toutes les descriptions d'appareil conformes à la présente norme.

Structure lexicale

EDD_VERSION integer

L'attribut EDD_VERSION est spécifié dans le Tableau 6.

Tableau 6 – Attribut EDD_VERSION

Utilisa-tion	Attribut	Description
o	integer	Entier non signé à deux octets contenant le chiffre 1 pour la première édition de la norme.

7.2.2.6 MANUFACTURER**Objet**

L'attribut MANUFACTURER identifie le fabricant. Il convient que l'allocation de code soit gérée par les organisations spécifiques qui sont responsables des différents profils EDDL. La combinaison d'un attribut EDD_PROFILE et d'un attribut MANUFACTURER doit être unique.

Structure lexicale

MANUFACTURER (long integer, identifier)

Les attributs MANUFACTURER sont spécifiés dans le Tableau 7.

Tableau 7 – Attributs MANUFACTURER

Utilisa-tion	Attribut	Description
s	long integer	Entier long non signé contenant le code pour un fabricant spécifique. Si tous les bits dans la valeur de l'entier sont définis à 1, la structure lexicale MANUFACTURER_EXT doit être présente et identifier le fabricant.
sm	identifier	Référence à un entier long non signé qui contient le code pour un fabricant spécifique.
Il convient que le mapping de l'identificateur avec une constante entière soit stocké dans un fichier séparé appelé "devices.cfg". Ce fichier contient une liste d'identificateurs pour les fabricants.		

7.2.2.7 MANUFACTURER_EXT**Objet**

L'attribut MANUFACTURER_EXT est réservé pour une utilisation future. C'est un espace réservé pour une chaîne permettant l'identification globale du MANUFACTURER.

NOTE La spécification de cet attribut sera incluse lorsqu'elle sera établie au niveau international.

Structure lexicale

MANUFACTURER_EXT string

L'attribut MANUFACTURER_EXT est spécifié dans le Tableau 8.

Tableau 8 – Attribut MANUFACTURER_EXT

Utilisa-tion	Attribut	Description
c	string	Chaîne contenant les informations pour l'identification mondiale unique d'un fabricant.

7.3 AXIS

7.3.1 Structure générale

Objet

AXIS décrit un axe d'un attribut CHART ou GRAPH.

Structure lexicale

AXIS *identifieur* [CONSTANT_UNIT , HELP, LABEL, MAX_VALUE, MIN_VALUE, SCALING]

Les attributs AXIS sont spécifiés dans le Tableau 9.

Tableau 9 – Attributs AXIS

Utilisa-tion	Attribut	Description
o	CONSTANT_UNIT	Elément lexical (voir 7.3.2.1).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	MAX_VALUE	Elément lexical (voir 7.3.2.2).
o	MIN_VALUE	Elément lexical (voir 7.3.2.3).
o	SCALING	Elément lexical (voir 7.3.2.4).

7.3.2 Attributs spécifiques

7.3.2.1 CONSTANT_UNIT

Objet

L'attribut CONSTANT_UNIT est utilisé, si AXIS a un code unitaire qui ne change jamais. L'attribut CONSTANT_UNIT est spécifié comme étant une chaîne qui sera affichée avec la valeur. Un attribut AXIS sans unité constante n'a pas d'unité associée ou bien les unités ne sont pas constantes.

Structure lexicale

CONSTANT_UNIT (string)<exp>

L'attribut CONSTANT_UNIT est spécifié dans le Tableau 176.

7.3.2.2 MAX_VALUE

Objet

L'attribut MAX_VALUE spécifie la valeur associée à la dernière marque de graduation d'AXIS. La dernière marque de graduation d'un attribut X_AXIS et d'un attribut Y_AXIS correspond respectivement à la marque de graduation la plus à droite et la plus haute. Par défaut, la valeur associée à la dernière marque de graduation d'un attribut AXIS sans attribut MAX_VALUE est déduite des attributs WAVEFORM sur GRAPH.

Structure lexicale

MAX_VALUE (reference, double, float, integer, unsigned_integer)<exp>

Les attributs MAX_VALUE sont spécifiés dans le Tableau 10.

Tableau 10 – Attributs MAX_VALUE, MIN_VALUE

Utilisa-tion	Attribut	Description
o	reference	Référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY ou un élément d'une instance de REFERENCE_ARRAY avec un type de données date/numérique. MAX_VALUE et MIN_VALUE doivent référencer des attributs VARIABLE du même type de données. Cela doit s'appliquer pour X_AXIS et Y_AXIS.
o	double	Constante à virgule flottante à double précision.
o	float	Constante à virgule flottante à simple précision.
o	integer	Constante entière.
o	unsigned_integer	Constante entière non signée.

7.3.2.3 MIN_VALUE**Objet**

L'attribut MIN_VALUE spécifie la valeur associée à la première marque de graduation d'AXIS. La première marque de graduation d'X_AXIS et d'Y_AXIS correspond respectivement à la marque de graduation la plus à gauche et la plus basse. Par défaut, la valeur associée à la première marque de graduation d'un attribut AXIS sans attribut MIN_VALUE est déduite des attributs WAVEFORM sur GRAPH.

Structure lexicale

MIN_VALUE (reference, double, float, integer, unsigned_integer) <exp>

Les attributs MIN_VALUE sont spécifiés dans le Tableau 10.

7.3.2.4 SCALING**Objet**

L'attribut SCALING spécifie la façon selon laquelle il convient que les marques de graduation d'AXIS soient mises à l'échelle. Par défaut, un AXIS sans attribut SCALING est mis à l'échelle de façon linéaire.

Structure lexicale

SCALING (LINEAR, LOGARITHMIC) <exp>

Les attributs SCALING sont spécifiés dans le Tableau 11.

Tableau 11 – Attributs SCALING

Utilisa-tion	Attribut	Description
s	LINEAR	Il convient que les marques de graduation sur l'axe soient mises à l'échelle de façon linéaire.
s	LOGARITHMIC	Il convient que les marques de graduation sur l'axe soient mises à l'échelle de façon logarithmique (base 10).

7.4 BLOCK

7.4.1 BLOCK_A

7.4.1.1 Structure générale

Objet

Un appareil peut être organisé en blocs logiques. Chaque attribut BLOCK_A est un groupe logique d'attributs PARAMETERS et apporte des informations additionnelles pour les applications BLOCK_A. Les arguments de l'attribut PROGRAM sont décrits par l'attribut CHARACTERISTICS. Plusieurs attributs BLOCK_A du même type peuvent être instanciés.

NOTE Cette approche de BLOCK_A est optimisée pour l'efficacité d'accès.

Structure lexicale

BLOCK_A identifieur [CHARACTERISTICS, LABEL, PARAMETERS, AXIS_ITEMS,
CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISPLAY_ITEMS, FILE_ITEMS,
GRAPH_ITEMS, GRID_ITEMS, HELP, IMAGE_ITEMS, LIST_ITEMS, MENU_ITEMS,
METHOD_ITEMS, PARAMETER_LISTS, PLUGIN_ITEMS, REFERENCE_ARRAY_ITEMS,
REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS,
WRITE_AS_ONE_ITEMS, CHARTS, FILES, LISTS, GRAPHS, GRIDS, MENUS,
METHODS, PLUGINS]

Les attributs BLOCK_A sont spécifiés dans le Tableau 12.

Tableau 12 – Attributs BLOCK_A

Utilisa-tion	Attribut	Description
m	CHARACTERISTICS	Elément lexical (voir 7.4.1.2.1).
m	LABEL	Elément lexical (voir 7.36.9).
m	PARAMETERS	Elément lexical (voir 7.4.1.2.2).
o	AXIS_ITEMS	Elément lexical (voir 7.4.1.2.3).
o	CHART_ITEMS	Elément lexical (voir 7.4.1.2.4).
o	COLLECTION_ITEMS	Elément lexical (voir 7.4.1.2.5).
o	EDIT_DISPLAY_ITEMS	Elément lexical (voir 7.4.1.2.6).
o	FILE_ITEMS	Elément lexical (voir 7.4.1.2.7).
o	GRAPH_ITEMS	Elément lexical (voir 7.4.1.2.8).
o	GRID_ITEMS	Elément lexical (voir 7.4.1.2.9).
o	HELP	Elément lexical (voir 7.36.8).
o	IMAGE_ITEMS	Elément lexical (voir 7.4.1.2.10).
o	LIST_ITEMS	Elément lexical (voir 7.4.1.2.11).
o	LOCAL_PARAMETERS	Elément lexical (voir 7.4.1.2.12).
o	MENU_ITEMS	Elément lexical (voir 7.4.1.2.13).
o	METHOD_ITEMS	Elément lexical (voir 7.4.1.2.14).
o	PARAMETER_LISTS	Elément lexical (voir 7.4.1.2.15).
o	PLUGIN_ITEMS	Elément lexical (voir 7.4.1.2.29).
o	REFERENCE_ARRAY_ITEMS	Elément lexical (voir 7.4.1.2.16).
o	REFRESH_ITEMS	Elément lexical (voir 7.4.1.2.17).
o	SOURCE_ITEMS	Elément lexical (voir 7.4.1.2.18).
o	UNIT_ITEMS	Elément lexical (voir 7.4.1.2.19).
o	WAVEFORM_ITEMS	Elément lexical (voir 7.4.1.2.20).
o	WRITE_AS_ONE_ITEMS	Elément lexical (voir 7.4.1.2.21).
o	CHARTS	Elément lexical (voir 7.4.1.2.22).
o	FILES	Elément lexical (voir 7.4.1.2.28).
o	LISTS	Elément lexical (voir 7.4.1.2.23).
o	GRAPHS	Elément lexical (voir 7.4.1.2.24).
o	GRIDS	Elément lexical (voir 7.4.1.2.25).
o	MENUS	Elément lexical (voir 7.4.1.2.26).
o	METHODS	Elément lexical (voir 7.4.1.2.27).
o	PLUGINS	Elément lexical (voir 7.4.1.2.30).

7.4.1.2 Attributs spécifiques

7.4.1.2.1 CHARACTERISTICS

Objet

L'attribut CHARACTERISTICS spécifie les caractéristiques externes de BLOCK_A. Il peut inclure la classe, le type de fonction, le type de bloc et le temps d'exécution. Les caractéristiques externes d'un attribut BLOCK_A sont contenues dans RECORD.

NOTE En général, les attributs CHARACTERISTICS de BLOCK_A fixent le type de BLOCK_A.

Structure lexicale

CHARACTERISTICS reference

L'attribut CHARACTERISTICS est spécifié dans le Tableau 13.

Tableau 13 – Attribut CHARACTERISTICS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de RECORD.

7.4.1.2.2 PARAMETERS

Objet

L'attribut PARAMETERS contient une liste de références à des instances de VARIABLE, VALUE_ARRAY ou RECORD. Chaque référence doit avoir un identificateur et peut avoir une description et/ou un texte d'aide.

NOTE Les attributs PARAMETERS dans BLOCK_A correspondent à des objets de communication individuels, répertoriés dans un dictionnaire d'objets de l'appareil. Les éléments PARAMETERS font référence aux définitions de communication de ces objets. Le dictionnaire d'objets localise les emplacements spécifiques des données dans un appareil.

Structure lexicale

PARAMETERS [identifier, reference, description, help]+

Les attributs PARAMETERS sont spécifiés dans le Tableau 14.

Tableau 14 – Attributs PARAMETERS

Utilisa-tion	Attribut	Description
m	identifier	Identificateur de VARIABLE, VALUE_ARRAY ou RECORD. Chaque élément de la liste des attributs PARAMETERS doit utiliser un identificateur dans l'EDD pour faire référence à celle-ci.
m	reference	Référence à une instance de VARIABLE, VALUE_ARRAY ou RECORD.
o	description	Brève description pour la référence.
o	help	Texte d'aide pour la référence.

7.4.1.2.3 AXIS_ITEMS

Objet

L'attribut AXIS_ITEMS est une liste des instances d'AXIS associées à BLOCK_A.

Structure lexicale

AXIS_ITEMS reference+

L'attribut AXIS_ITEMS est spécifié dans le Tableau 15.

Tableau 15 – Attribut AXIS_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance d'AXIS.

7.4.1.2.4 CHART_ITEMS**Objet**

L'attribut CHART_ITEMS est une liste des instances de CHART associées à BLOCK_A.

Structure lexicale

CHART_ITEMS reference+

L'attribut CHART_ITEMS est spécifié dans le Tableau 16.

Tableau 16 – Attribut CHART_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de CHART.

7.4.1.2.5 COLLECTION_ITEMS**Objet**

L'attribut COLLECTION_ITEMS est une liste des instances de COLLECTION associées à BLOCK_A.

Structure lexicale

COLLECTION_ITEMS reference+

L'attribut COLLECTION_ITEMS est spécifié dans le Tableau 17.

Tableau 17 – Attribut COLLECTION_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de COLLECTION.

7.4.1.2.6 EDIT_DISPLAY_ITEMS**Objet**

L'attribut EDIT_DISPLAY_ITEMS est une liste des instances d'EDIT_DISPLAY associées à BLOCK_A.

Structure lexicale

EDIT_DISPLAY_ITEMS reference+

L'attribut EDIT_DISPLAY_ITEMS est spécifié dans le Tableau 18.

Tableau 18 – Attribut EDIT_DISPLAY_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance d'EDIT_DISPLAY.

7.4.1.2.7 FILE_ITEMS**Objet**

L'attribut FILE_ITEMS est une liste des instances de FILE associées à BLOCK_A.

Structure lexicale

FILE_ITEMS reference+

L'attribut FILE_ITEMS est spécifié dans le Tableau 19.

Tableau 19 – Attribut FILE_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de FILE.

7.4.1.2.8 GRAPH_ITEMS**Objet**

L'attribut GRAPH_ITEMS est une liste des instances de GRAPH associées à BLOCK_A.

Structure lexicale

GRAPH_ITEMS reference+

L'attribut GRAPH_ITEMS est spécifié dans le Tableau 20.

Tableau 20 – Attribut GRAPH_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de GRAPH.

7.4.1.2.9 GRID_ITEMS**Objet**

L'attribut GRID_ITEMS est une liste d'instances de GRID associées à BLOCK_A.

Structure lexicale

GRID_ITEMS reference+

L'attribut GRID_ITEMS est spécifié dans le Tableau 21.

Tableau 21 – Attribut GRID_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de GRID.

7.4.1.2.10 IMAGE_ITEMS**Objet**

L'attribut IMAGE_ITEMS est une liste des instances d'IMAGE associées à BLOCK_A.

Structure lexicale

IMAGE_ITEMS reference+

L'attribut IMAGE_ITEMS est spécifié dans le Tableau 22.

Tableau 22 – Attribut IMAGE_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance d'IMAGE.

7.4.1.2.11 LIST_ITEMS**Objet**

L'attribut LIST_ITEMS est une liste des instances de LIST associées à BLOCK_A.

Structure lexicale

LIST_ITEMS reference+

L'attribut LIST_ITEMS est spécifié dans le Tableau 23.

Tableau 23 – Attribut LIST_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de LIST.

7.4.1.2.12 LOCAL_PARAMETERS**Objet**

L'attribut LOCAL_PARAMETERS contient une liste de références à des instances de VARIABLE, VALUE_ARRAY ou RECORD. Chaque référence doit avoir un identificateur et peut avoir une description et/ou un texte d'aide.

NOTE Les attributs LOCAL_PARAMETERS dans BLOCK_A ne correspondent pas à des objets de communication; ce sont des paramètres qui sont utilisés uniquement par l'application EDD.

Structure lexicale

LOCAL_PARAMETERS [identifier, reference, description, help] +

Les attributs LOCAL_PARAMETERS sont spécifiés dans le Tableau 14.

7.4.1.2.13 MENU_ITEMS

Objet

L'attribut MENU_ITEMS spécifie les attributs MENU associés à BLOCK_A.

Structure lexicale

`MENU_ITEMS reference+`

L'attribut MENU_ITEMS est spécifié dans le Tableau 24.

Tableau 24 – Attribut MENU_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à un MENU.

7.4.1.2.14 METHOD_ITEMS

Objet

L'attribut METHOD_ITEMS spécifie les attributs METHOD associés à BLOCK_A.

Structure lexicale

`METHOD_ITEMS reference+`

L'attribut METHOD_ITEMS est spécifié dans le Tableau 25.

Tableau 25 – Attribut METHOD_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de METHOD.

7.4.1.2.15 PARAMETER_LISTS

Objet

L'attribut PARAMETER_LISTS spécifie les attributs VARIABLE_LIST associés à un attribut BLOCK_A, ainsi qu'une description facultative et une aide pour chaque VARIABLE_LIST. Chaque attribut BLOCK_A peut contenir plusieurs attributs PARAMETER_LISTS.

Structure lexicale

`PARAMETER_LISTS [identifiant, reference, description, help] +`

Les attributs PARAMETER_LISTS sont spécifiés dans le Tableau 26.

Tableau 26 – Attributs PARAMETER_LISTS

Utilisa-tion	Attribut	Description
m	identifiant	Identificateur de VARIABLE_LIST. Chaque élément de la liste d'attributs PARAMETER_LIST doit utiliser un identificateur dans l'EDD pour faire référence à celle-ci.
m	reference	Référence à une instance de VARIABLE_LIST.
o	description	Brève description de l'élément.
o	help	Texte d'aide pour l'élément.

7.4.1.2.16 REFERENCE_ARRAY_ITEMS**Objet**

L'attribut REFERENCE_ARRAY_ITEMS spécifie les attributs REFERENCE_ARRAY associés à BLOCK_A.

Structure lexicale

REFERENCE_ARRAY_ITEMS reference+

L'attribut REFERENCE_ARRAY_ITEMS est spécifié dans le Tableau 27.

Tableau 27 – Attribut REFERENCE_ARRAY_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de REFERENCE_ARRAY.

7.4.1.2.17 REFRESH_ITEMS**Objet**

L'attribut REFRESH_ITEMS spécifie les relations REFRESH associées à BLOCK_A.

Structure lexicale

REFRESH_ITEMS reference+

L'attribut REFRESH_ITEMS est spécifié dans le Tableau 28.

Tableau 28 – Attribut REFRESH_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de REFRESH.

7.4.1.2.18 SOURCE_ITEMS**Objet**

L'attribut SOURCE_ITEMS spécifie les instances de SOURCE associées à BLOCK_A.

Structure lexicale

SOURCE_ITEMS reference+

L'attribut SOURCE_ITEMS est spécifié dans le Tableau 29.

Tableau 29 – Attribut SOURCE_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de SOURCE.

7.4.1.2.19 UNIT_ITEMS

Objet

L'attribut UNIT_ITEMS spécifie les relations des attributs UNIT associées à BLOCK_A.

Structure lexicale

UNIT_ITEMS reference+

L'attribut UNIT_ITEMS est spécifié dans le Tableau 30.

Tableau 30 – Attribut UNIT_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance d'UNIT.

7.4.1.2.20 WAVEFORM_ITEMS

Objet

L'attribut WAVEFORM_ITEMS spécifie les instances de WAVEFORM associées à BLOCK_A.

Structure lexicale

WAVEFORM_ITEMS reference+

L'attribut WAVEFORM_ITEMS est spécifié dans le Tableau 31.

Tableau 31 – Attribut WAVEFORM_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de WAVEFORM.

7.4.1.2.21 WRITE_AS_ONE_ITEMS

Objet

L'attribut WRITE_AS_ONE_ITEMS spécifie les relations WRITE_AS_ONE associées à BLOCK_A.

Structure lexicale

WRITE_AS_ONE_ITEMS reference+

L'attribut WRITE_AS_ONE_ITEMS est spécifié dans le Tableau 32.

Tableau 32 – Attribut WRITE_AS_ONE_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de WRITE_AS_ONE.

7.4.1.2.22 CHARTS

Objet

L'attribut CHARTS spécifie les instances de CHART associées à BLOCK_A. Seuls les attributs CHART qui seront référencés via une instance de BLOCK_A spécifique (voir 7.38.21) doivent être spécifiés.

Structure lexicale

CHARTS (identifiant, reference, description, help)+

Les attributs CHARTS sont spécifiés dans le Tableau 33.

Tableau 33 – Attributs CHARTS

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel CHART peut être référencé.
m	reference	Référence à une instance de CHART.
o	description	Brève description de CHART.
o	help	Texte d'aide pour l'attribut CHART.

7.4.1.2.23 LISTS

Objet

L'attribut LISTS spécifie les instances de LIST associées à BLOCK_A. Seuls les attributs LIST qui seront référencés via une instance de BLOCK_A spécifique (voir 7.4.1) doivent être spécifiés.

Structure lexicale

LISTS (identifiant, reference, description, help)+

Les attributs LISTS sont spécifiés dans le Tableau 34.

Tableau 34 – Attributs LISTS

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel LIST peut être référencé.
m	reference	Référence à une instance de LIST.
o	description	Brève description de LIST.
o	help	Texte d'aide pour l'attribut LIST.

7.4.1.2.24 GRAPHS

Objet

L'attribut GRAPHS spécifie les instances de GRAPH associées à BLOCK_A. Seuls les attributs GRAPH qui seront référencés via une instance de BLOCK_A spécifique (voir 7.38.23) doivent être spécifiés.

Structure lexicale

GRAPHS (identifiant, reference, description, help)+

Les attributs GRAPHS sont spécifiés dans le Tableau 35.

Tableau 35 – Attributs GRAPHS

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel GRAPH peut être référencé.
m	reference	Référence à une instance de GRAPH.
o	description	Brève description de GRAPH.
o	help	Texte d'aide pour l'attribut GRAPH.

7.4.1.2.25 GRIDS

Objet

L'attribut GRIDS spécifie les instances de GRID associées à BLOCK_A. Seuls les attributs GRID qui seront référencés via une instance de BLOCK_A spécifique (voir 7.38.24) doivent être spécifiés.

Structure lexicale

GRIDS (identifiant, reference, description, help)+

Les attributs GRIDS sont spécifiés dans le Tableau 36.

Tableau 36 – Attributs GRIDS

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel GRID peut être référencé.
m	reference	Référence à une instance de GRID.
o	description	Brève description de GRID.
o	help	Texte d'aide pour l'attribut GRID.

7.4.1.2.26 MENUS

Objet

L'attribut MENUS spécifie les instances de MENU associées à BLOCK_A. Seuls les attributs MENU qui seront référencés via une instance de BLOCK_A spécifique (voir 7.38.25) doivent être spécifiés.

Structure lexicale

MENUS (identifiant, reference, description, help)+

Les attributs MENUS sont spécifiés dans le Tableau 37.

Tableau 37 – Attributs MENUS

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel MENU peut être référencé.
m	reference	Référence à une instance de MENU.
o	description	Brève description de MENU.
o	help	Texte d'aide pour l'attribut MENU.

7.4.1.2.27 METHODS

Objet

L'attribut METHODS spécifie les instances de METHOD associées à BLOCK_A. Seuls les attributs METHOD qui seront référencés via une instance de BLOCK_A spécifique (voir 7.38.26) doivent être spécifiés.

Structure lexicale

METHODS (identifiant, reference, description, help)+

Les attributs METHODS sont spécifiés dans le Tableau 38.

Tableau 38 – Attributs METHODS

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel METHOD peut être référencé.
m	reference	Référence à une instance de METHOD.
o	description	Brève description de METHOD.
o	help	Texte d'aide pour METHOD.

7.4.1.2.28 FILES

Objet

L'attribut FILES spécifie les instances de FILE associées à BLOCK_A. Seuls les attributs FILE qui seront référencés via une instance de BLOCK_A spécifique (voir 7.38.29) doivent être spécifiés.

Structure lexicale

FILES (identifiant, reference, description, help)+

Les attributs FILES sont spécifiés dans le Tableau 39.

Tableau 39 – Attributs FILES

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel FILE peut être référencé.
m	reference	Référence à une instance de FILE.
o	description	Brève description de FILE.
o	help	Texte d'aide pour FILE.

7.4.1.2.29 PLUGIN_ITEMS**Objet**

L'attribut PLUGIN_ITEMS spécifie les attributs PLUGIN associés à BLOCK_A.

Structure lexicale

PLUGIN_ITEMS reference+

L'attribut PLUGIN_ITEMS est spécifié dans le Tableau 40.

Tableau 40 – Attribut PLUGIN_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de PLUGIN.

7.4.1.2.30 PLUGINS**Objet**

L'attribut PLUGINS spécifie les attributs PLUGIN associés à BLOCK_A. Seuls les attributs PLUGIN qui seront référencés via une instance de BLOCK_A spécifique (voir 7.38.30) doivent être spécifiés.

Structure lexicale

PLUGINS (identifiant, reference, description, help)+

Les attributs PLUGINS sont spécifiés dans le Tableau 41.

Tableau 41 – Attributs PLUGINS

Utilisa-tion	Attribut	Description
m	identifiant	Nom par lequel PLUGIN peut être référencé.
m	reference	Référence à une instance de PLUGIN.
o	description	Brève description de PLUGIN.
o	help	Texte d'aide pour l'attribut PLUGIN.

7.4.2 BLOCK_B**7.4.2.1 Structure générale****Objet**

Les variables d'un appareil ou module sont respectivement structurées en blocs correspondant aux composants ou aux parties fonctionnelles d'un appareil. Trois types de blocs sont définis: PHYSICAL, TRANSDUCER et FUNCTION. Plusieurs BLOCK_B du même TYPE peuvent être instanciés. Afin de faciliter l'accès, chaque instance a son propre attribut NUMBER. Pour chacun des types de blocs, il existe différentes chaînes de séquences de NUMBER, commençant par 1.

Lors de l'accès à un attribut VARIABLE dans un appareil, la construction de BLOCK_B est utilisée conjointement avec COMMAND pour permettre un adressage relatif.

NOTE 1 Cette approche basée sur BLOCK_B est optimisée pour une meilleure efficacité de la mémoire.

NOTE 2 Le stockage d'une variable dans un appareil est spécifique au fabricant et est représenté par un répertoire d'appareils. Un répertoire d'appareils contient un résumé des entrées de BLOCK_B disponibles dans un appareil. De plus, le répertoire d'appareils contient pour chaque attribut BLOCK_B un pointeur vers son adresse physique. Une application EDD trouve un objet unique en ajoutant un décalage à l'adresse physique de BLOCK_B.

NOTE 3 Dans une instance de BLOCK_B, un indexage relatif est utilisé. Cet index relatif est défini dans le profil d'appareil.

Structure lexicale

BLOCK_B *identifier* [NUMBER, TYPE]

Les attributs BLOCK_B sont spécifiés dans le Tableau 42.

Tableau 42 – Attributs BLOCK_B

Utilisa-tion	Attribut	Description
m	NUMBER	Élément lexical (voir 7.4.2.2.1).
m	TYPE	Élément lexical (voir 7.4.2.2.2).

7.4.2.2 Attributs spécifiques

7.4.2.2.1 NUMBER

Objet

L'attribut NUMBER énumère chaque attribut BLOCK_B du même TYPE dans l'appareil par un nombre entier supérieur à zéro. Un appareil peut contenir plusieurs instances de BLOCK_B, qui sont identifiées avec l'attribut NUMBER.

Structure lexicale

NUMBER (*integer*, *expression*)<exp>

Les attributs NUMBER sont spécifiés dans le Tableau 43.

Tableau 43 – Attributs NUMBER

Utilisa-tion	Attribut	Description
s	integer	Numéro de l'instance de BLOCK_B du même TYPE.
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut NUMBER (pour la description des expressions, voir 7.40).

7.4.2.2.2 TYPE

Objet

L'attribut TYPE est utilisé pour spécifier un des trois types de blocs.

Structure lexicale

TYPE [PHYSICAL, TRANSDUCER, FUNCTION]

Les attributs TYPE sont spécifiés dans le Tableau 44.

Tableau 44 – Attributs TYPE

Utilisa-tion	Attribut	Description
s	FUNCTION	Bloc nommé constitué d'un ou plusieurs paramètres d'entrée, de sortie ou internes. Les fonctions blocs représentent les fonctions d'automatisation de base exécutées par une application EDD, qui est indépendante des appareils spécifiques et du réseau. Chaque fonction bloc traite des paramètres d'entrée conformément à un algorithme spécifié et un ensemble de paramètres internes. Ils génèrent des paramètres de sortie qui sont disponibles pour utilisation dans la même application EDD de fonction bloc ou par d'autres applications de fonction bloc.
s	PHYSICAL	Les caractéristiques spécifiques du matériel/de la ressource d'un appareil sont rendues visibles par l'intermédiaire du bloc physique. De manière similaire aux blocs transducteur, ils isolent des fonctions blocs du matériel physique en contenant un ensemble de paramètres du matériel/ressource indépendants de l'implémentation.
s	TRANSDUCER	Les blocs transducteur isolent les fonctions blocs des éléments spécifiques des appareils E/S, tels que des capteurs, des actionneurs et des commutateurs. Les blocs transducteur contrôlent l'accès à des appareils E/S par l'intermédiaire d'une interface indépendante des appareils définie pour une utilisation par des fonctions blocs. Les blocs transducteur remplissent des fonctions, telles que l'étalonnage et la linéarisation, sur des données E/S pour convertir celles-ci en représentation indépendante de l'appareil.

7.5 CHART

7.5.1 Structure générale

Objet

CHART décrit un diagramme utilisé pour afficher des données provenant d'un appareil dans l'application EDD. Un attribut CHART est utilisé pour afficher des valeurs de données continues provenant de l'appareil, par opposition à un attribut GRAPH qui est utilisé pour afficher un ensemble de données qui est stocké dans un appareil. Les données provenant de l'appareil peuvent être ajustées avant d'être affichées via l'utilisation d'attributs INIT_ACTIONS et REFRESH_ACTIONS sur les membres de SOURCE de l'attribut CHART.

Structure lexicale

CHART identifiant [MEMBERS, CYCLE_TIME, HEIGHT, WIDTH, LENGTH, HELP, LABEL, TYPE, VALIDITY, VISIBILITY]

Les attributs CHART sont spécifiés dans le Tableau 45.

Tableau 45 – Attributs CHART

Utilisa-tion	Attribut	Description
M	MEMBERS	Elément lexical (voir 7.36.12).
o	CYCLE_TIME	Elément lexical (voir 7.5.2.1).
o	HEIGHT	Elément lexical (voir 7.36.7).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	LENGTH	Elément lexical (voir 7.5.2.2).
o	TYPE	Elément lexical (voir 7.5.2.3).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	WIDTH	Elément lexical (voir 7.36.17).

7.5.2 Attributs spécifiques

7.5.2.1 CYCLE_TIME

Objet

L'attribut CYCLE_TIME spécifie le temps, en ms, durant lequel l'application EDD doit attendre entre des lectures successives des valeurs sources de l'appareil. Par défaut, la durée de cycle d'un attribut CHART sans attribut CYCLE_TIME est de 1 000 ms.

Structure lexicale

CYCLE_TIME (integer)<exp>

L'attribut CYCLE_TIME est spécifié dans le Tableau 46.

Tableau 46 – Attribut CYCLE_TIME

Utilisa-tion	Attribut	Description
m	integer	Temps, en ms, entre des lectures successives des valeurs sources de l'appareil. Après que CHART est affiché par une application EDD, CYCLE_TIME ne doit pas changer.

7.5.2.2 LENGTH

Objet

L'attribut LENGTH spécifie la durée, en ms, qui est affichée par l'attribut CHART. Le nombre d'échantillons affichés par un diagramme peut être calculé en divisant l'attribut LENGTH par l'attribut CYCLE_TIME. Par défaut, la durée de cycle d'un attribut CHART sans attribut LENGTH est de 600 000 ms.

Structure lexicale

LENGTH (integer)<exp>

L'attribut LENGTH est spécifié dans le Tableau 47.

Tableau 47 – Attribut LENGTH

Utilisa-tion	Attribut	Description
m	integer	Temps, en ms, qui est affiché par l'attribut CHART. Après que CHART est affiché par une application EDD, LENGTH ne doit pas changer.

7.5.2.3 TYPE

Objet

L'attribut TYPE spécifie le type de CHART que l'EDD doit afficher. Par défaut, le type d'un attribut CHART sans attribut TYPE est STRIP.

Structure lexicale

TYPE (GAUGE, HORIZONTAL_BAR, SCOPE, STRIP, SWEEP, VERTICAL_BAR) <exp>

Les attributs TYPE sont spécifiés dans le Tableau 48.

Tableau 48 – Attributs TYPE

Utilisa-tion	Attribut	Description
s	GAUGE	Une valeur source unique est affichée sous forme de jauge, qui est une représentation graphique des données similaire à la jauge de carburant d'une automobile.
s	HORIZONTAL_BAR	Les valeurs sources sont affichées sous la forme d'un diagramme à barres horizontal.
s	SCOPE	Les données de ce diagramme sont mises à jour de gauche à droite. Lorsque les valeurs sources atteignent la fin de la zone d'affichage de CHART, la zone d'affichage est effacée et les nouvelles valeurs sources sont à nouveau affichées à partir du début de la zone d'affichage. L'axe X doit être mis à jour correctement en fonction de la zone d'affichage.
s	STRIP	Les données de ce diagramme sont mises à jour de gauche à droite. Lorsque les valeurs sources atteignent la fin de la zone d'affichage de CHART, la zone d'affichage est décalée, les valeurs sources les plus anciennes sont effacées de la zone d'affichage et les nouvelles valeurs sources sont ajoutées à la zone d'affichage. L'axe X doit être parcouru et mis à jour correctement en fonction de la zone d'affichage.
s	SWEEP	Les données de ce diagramme sont mises à jour de gauche à droite. Lorsque les valeurs sources atteignent la fin de la zone d'affichage de CHART, les nouvelles valeurs sources sont affichées au début de la zone d'affichage, mais contrairement à SCOPE, seule la partie de la zone d'affichage nécessaire pour afficher les nouvelles valeurs sources est effacée. L'axe X doit être parcouru et mis à jour correctement en fonction de la zone d'affichage.
s	VERTICAL_BAR	Les valeurs sources sont affichées sous la forme d'un diagramme à barres vertical.

7.6 COLLECTION

7.6.1 Structure générale

Objet

COLLECTION est un groupe logique d'éléments, tels que des attributs VARIABLE ou MENU. Un identificateur est assigné à chaque élément. Les éléments peuvent être référencés dans la description de l'appareil en utilisant l'identificateur de COLLECTION et le nom de l'élément.

COLLECTION est destiné à être utilisé pour identifier des paramètres qu'il convient de traiter conjointement, par exemple. Il convient que l'application EDD prenne en charge au minimum l'imbrication de deux niveaux, par exemple COLLECTION de COLLECTION.

Structure lexicale

COLLECTION `identifieur, item-type, [MEMBERS, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]`

Les attributs COLLECTION sont spécifiés dans le Tableau 49.

Tableau 49 – Attributs COLLECTION

Utilisa-tion	Attribut	Description
o	item-type	Elément lexical (voir 7.6.2).
m	MEMBERS	Elément lexical (voir 7.36.12).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	PRIVATE	Elément lexical (voir 7.36.18).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).

7.6.2 Attribut spécifique – item-type

Objet

L'attribut item-type spécifie le type de membres dans la collection. Si item-type est spécifié, tous les membres de collection doivent être du type spécifié. Sinon, les membres de collection peuvent être d'un type quelconque.

Structure lexicale

`item-type`

Le Tableau 50 présente les attributs item-type admis.

Tableau 50 – item-type

Utilisa-tion	item-type	Description
s	BLOCK_A	Sélection de cette construction de base.
s	BLOCK_B	Sélection de cette construction de base.
s	CHART	Sélection de cette construction de base.
s	COLLECTION	Sélection de cette construction de base.
s	EDIT_DISPLAY	Sélection de cette construction de base.
s	FILE	Sélection de cette construction de base.
s	GRAPH	Sélection de cette construction de base.
s	GRID	Sélection de cette construction de base.
s	IMAGE	Sélection de cette construction de base.
s	LIST	Sélection de cette construction de base.
s	MENU	Sélection de cette construction de base.
s	METHOD	Sélection de cette construction de base.
s	PLUGIN	Sélection de cette construction de base.
s	RECORD	Sélection de cette construction de base.
s	REFERENCE_ARRAY	Sélection de cette construction de base.
s	VALUE_ARRAY	Sélection de cette construction de base.
s	VARIABLE	Sélection de cette construction de base.
s	VARIABLE_LIST	Sélection de cette construction de base.

7.7 COMMAND

7.7.1 Structure générale

Objet

Un attribut COMMAND est une construction utilisée pour prendre en charge le mapping des éléments de communication EDD avec un système de communication choisi. Il spécifie différents éléments exigés pour construire une trame de communication. Si cette construction est utilisée par un profil EDDL, chaque élément de données à transmettre doit être référencé à l'intérieur d'un attribut COMMAND ainsi que dans l'attribut MENU de transmission/téléchargement.

NOTE 1 Le champ d'adresse d'une trame de communication est spécifié à l'aide des constructions sélectionnables BLOCK, SLOT, SUB_SLOT, NUMBER, INDEX, MODULE, comme applicable. Le type de trame de communication ou son champ de contrôle est spécifié par la construction OPERATION. Le contenu de données de la trame de communication est spécifié par la construction TRANSACTION.

Structure lexicale

COMMAND *identif*ier, [OPERATION, TRANSACTION+, INDEX, BLOCK_B, NUMBER, SLOT, SUB_SLOT, API, HEADER, POST_RQSTRECEIVE_ACTIONS, RESPONSE_CODES]

Les attributs COMMAND sont spécifiés dans le Tableau 51.

Tableau 51 – Attributs COMMAND

Utilisa-tion	Attribut	Description
m	OPERATION	Elément lexical (voir 7.7.2.1).
m	TRANSACTION	Elément lexical (voir 7.7.2.2).
c	INDEX	Elément lexical (voir 7.7.2.3).
s	BLOCK_B	Elément lexical (voir 7.7.2.4).
s	NUMBER	Elément lexical (voir 7.7.2.5).
s	SLOT	Elément lexical (voir 7.7.2.6).
s	SUB_SLOT	Elément lexical (voir 7.7.2.7).
o	API	Elément lexical (voir 7.7.2.11).
o	HEADER	Elément lexical (voir 7.7.2.9).
o	POST_RQSTRECEIVE_ACTIONS	Elément lexical (voir 7.7.2.12).
o	RESPONSE_CODES	Elément lexical (voir 7.36.14).

NOTE 2 Les codes de réponse apparaissant dans une transaction s'appliquent uniquement à cette transaction.

NOTE 3 Les codes de réponse apparaissant à l'extérieur d'une transaction quelconque s'appliquent à toutes les transactions.

NOTE 4 Si le même code de réponse est spécifié à la fois à l'intérieur et à l'extérieur d'une transaction, le code de réponse spécifié dans la transaction est prioritaire, mais uniquement pour cette transaction.

7.7.2 Attributs spécifiques

7.7.2.1 OPERATION

Objet

L'attribut OPERATION spécifie les actions effectuées par l'appareil lors de la réception d'une demande de service provenant du système de communication.

Structure lexicale

OPERATION [(COMMAND, READ, WRITE)]

Les attributs OPERATION sont spécifiés dans le Tableau 52.

Tableau 52 – Attributs OPERATION

Utilisa-tion	Attribut	Description
s	COMMAND	L'appareil effectue un ensemble prédéfini d'actions (par exemple, auto-essai, réinitialisation principale).
s	READ	Spécifie une transaction de lecture. L'appareil retourne les valeurs actuelles des variables spécifiées. VARIABLES doit être défini dans la section REPLY de TRANSACTION.
s	WRITE	Spécifie une transaction d'écriture. L'appareil reçoit les valeurs actuelles des variables spécifiées. VARIABLES doit être défini dans la section REQUEST de TRANSACTION.

7.7.2.2 TRANSACTION

7.7.2.2.1 Structure générale

Objet

Les transactions spécifient le champ de données des messages de demande et de réponse d'une commande. Plusieurs transactions peuvent être définies. Il convient que la syntaxe prenne en charge une identification unique de l'attribut TRANSACTION. Chaque attribut TRANSACTION peut comprendre un ensemble de codes de réponse qui s'applique seulement à cette transaction particulière.

EXEMPLE Des commandes de bloc, telles que les commandes 4 et 5 de HART⁸ Universal Command version 4 ou supérieure, sont des exemples de commandes de transaction multiples.

Structure lexicale

TRANSACTION [REPLY, REQUEST, integer, RESPONSE_CODES]

Les attributs TRANSACTION sont spécifiés dans le Tableau 53.

Tableau 53 – Attributs TRANSACTION

Utilisa-tion	Attribut	Description
m	REPLY	Élément lexical (voir 7.7.2.2.2.1).
m	REQUEST	Élément lexical (voir 7.7.2.2.2.2).
o	integer	Nombre d'attributs TRANSACTIONS (si plusieurs attributs TRANSACTION sont utilisés).
o	RESPONSE_CODES	Élément lexical (voir 7.36.14).

7.7.2.2.2 Attributs spécifiques

7.7.2.2.2.1 REPLY

Objet

L'attribut REPLY contient une liste d'éléments de données (constantes ou références à des variables, listes, matrices, ou collections), qui sont reçues depuis l'appareil. L'ordre de la liste doit être maintenu dans le champ de données correspondant dans le PDU communiqué. Si les données référencées ne sont pas de type VARIABLE, les données référencées à l'intérieur des éléments EDD doivent toujours se terminer sous forme de références de VARIABLE. Tous les autres types d'éléments EDD référencés ne sont pas admis. La combinaison de liste, matrice et collection avec un attribut item-mask n'est pas admise.

Lorsqu'un attribut VARIABLE ne commence et ne termine pas sur des limites d'octet, il convient qu'un masque soit utilisé pour spécifier comment la variable est encapsulée dans le champ de données.

Chaque bit dans le masque peut avoir une signification sémantique. De plus, si le bit le moins significatif est défini dans un attribut item-mask, l'élément de données suivant (le cas échéant) est contenu dans le ou les octet(s) suivant(s) du PDU. Si le bit le moins significatif est nul (non défini), l'élément de données suivant est contenu dans le(s) même(s) octet(s) du PDU.

⁸ Voir Annexe D.4.

Lorsqu'une liste est référencée, l'application EDD doit réinitialiser l'attribut LIST puis doit remplir l'attribut LIST avec les données du PDU reçu.

Même si le bit le moins significatif n'est pas utilisé pour des données réelles, il convient qu'il soit défini dans un item-mask (associé à une donnée "fictive") si des données suivent.

Structure lexicale

REPLY [integer, reference [item-mask, INFO, INDEX]]+

Les attributs REPLY sont spécifiés dans le Tableau 54.

Tableau 54 – Attributs REPLY et REQUEST

Utilisa-tion	Attribut	Description
s	integer	Entier littéral qui peut avoir plusieurs tailles.
s	reference	Référence à une instance de VARIABLE, ARRAY, COLLECTION, REFERENCE_ARRAY ou LIST.
o	item-mask	Motif de bits pour extraire l'attribut VARIABLE du champ de données. L'attribut item-mask doit être utilisé uniquement pour les types de données INTEGER, UNSIGNED INTEGER, ENUMERATED et BIT_ENUMERATED. La longueur du masque d'élément doit être de 1 octet à 4 octets et doit correspondre à la longueur de la donnée dans la liste de REQUEST ou REPLY.
o	INFO	VARIABLE n'est pas réellement stocké dans l'appareil. Elle contient des informations additionnelles pour assurer un traitement correct.
o	INDEX	VARIABLE est utilisé dans les attributs REQUEST ou REPLY en tant qu'index dans REFERENCE_ARRAY.

Une variable peut être qualifiée à la fois par les attributs INDEX et INFO, auquel cas elle est appelée variable d'index local. Les commandes avec des indices locaux se comportent exactement de la même façon que les commandes avec des variables qualifiées par INDEX à l'exception du fait que les variables d'index ne sont pas stockées dans l'appareil.

INFO peut être utilisé pour envoyer ou lire un attribut VARIABLE avec des unités différentes de celles utilisées dans l'appareil.

7.7.2.2.2 REQUEST

Objet

L'attribut REQUEST contient une liste de données (constantes ou références à des variables, listes, matrices, ou collections), qui sont envoyées à l'appareil. L'ordre de la liste doit être maintenu dans le champ de données correspondant dans le PDU communiqué. Si les données référencées ne sont pas de type VARIABLE, les données référencées à l'intérieur des éléments EDD doivent toujours se terminer sous forme de références de VARIABLE. Tous les autres types d'éléments EDD référencés ne sont pas admis. La combinaison de liste, matrice et collection avec un attribut item-mask n'est pas admise.

Lorsqu'une donnée ne commence et ne termine pas sur des limites d'octet du PDU, un attribut item-mask doit être utilisé pour spécifier comment l'élément de données est encapsulé dans le champ de données. L'attribut item-mask est un nombre entier d'octets qui doit être logiquement soumis à une opération AND avec le PDU communiqué pour extraire la donnée souhaitée.

Si aucun item-mask de données n'est spécifié, un masque implicite est utilisé avec tous les bits définis pour chaque octet, correspondant à la longueur de donnée.

Chaque bit dans le masque peut avoir une signification sémantique. De plus, si le bit le moins significatif est défini dans un item-mask, la donnée suivante (le cas échéant) est contenue dans le ou les octet(s) suivant(s) du PDU. Si le bit le moins significatif est nul (non défini), la donnée suivante est contenue dans le(s) même(s) octet(s) du PDU.

Même si le bit le moins significatif n'est pas utilisé pour des données réelles, il convient qu'il soit défini dans un item-mask (associé à une donnée "fictive") si des données suivent.

Structure lexicale

```
REQUEST [integer, reference [item-mask, INFO, INDEX]]+
```

Les attributs REQUEST sont spécifiés dans le Tableau 54.

7.7.2.3 INDEX

Objet

L'attribut INDEX doit être présent si l'attribut SLOT ou BLOCK_B est utilisé. INDEX spécifie l'adresse d'élément de données dans SLOT ou dans BLOCK_B.

Structure lexicale

```
INDEX (integer, reference, expression)<exp>
```

Les attributs INDEX sont spécifiés dans le Tableau 55.

Tableau 55 – Attributs INDEX

Utilisa-tion	Attribut	Description
s	integer	Numéro de l'index.
s	reference	Référence à une instance de VARIABLE contenant l'index.
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut INDEX (pour la description des expressions, voir 7.40).

7.7.2.4 BLOCK_B

Objet

L'attribut BLOCK_B fournit une référence à un bloc individuel dans l'appareil.

Structure lexicale

```
BLOCK_B (reference)<exp>
```

L'attribut BLOCK_B est spécifié dans le Tableau 56.

Tableau 56 – Attribut BLOCK_B

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_B.

7.7.2.5 NUMBER

Objet

L'attribut NUMBER est utilisé pour énumérer les attributs COMMAND.

Structure lexicale

NUMBER unsigned_integer

L'attribut NUMBER est spécifié dans le Tableau 57.

Tableau 57 – Attribut NUMBER

Utilisa- tion	Attribut	Description
m	unsigned_integer	Numéro de COMMAND.

7.7.2.6 SLOT

Objet

L'attribut SLOT spécifie le numéro d'une plage. Les données d'un appareil peuvent être allouées à des plages logiques. Dans une plage, chaque donnée est adressée en utilisant un index.

Structure lexicale

SLOT (integer, reference, expression)<exp>

Les attributs SLOT sont spécifiés dans le Tableau 58.

Tableau 58 – Attributs SLOT

Utilisa- tion	Attribut	Description
s	integer	Numéro de plage.
s	reference	Référence à une instance de VARIABLE contenant le numéro de plage.
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut SLOT (pour la description des expressions, voir 7.40).

7.7.2.7 SUB_SLOT

Objet

L'attribut SUB_SLOT spécifie le numéro d'une sous-plage. Les données d'un appareil peuvent être allouées à des sous-plages logiques. Dans une sous-plage, chaque donnée est adressée en utilisant un index.

Structure lexicale

SUB_SLOT (integer, reference, expression)<exp>

Les attributs SUB_SLOT sont spécifiés dans le Tableau 59.

Tableau 59 – Attributs SUB_SLOT

Utilisation	Attribut	Description
s	integer	Numéro de sous-plage.
s	reference	Référence à une instance de VARIABLE contenant le numéro de sous-plage.
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut SUB_SLOT (pour la description des expressions, voir 7.40).

7.7.2.8 CONNECTION (vide)**7.7.2.9 HEADER****Objet**

L'attribut HEADER est utilisé pour des protocoles de communication spécifiques.

Il peut être utilisé pour adresser des données dans un appareil ou pour une opération de réinitialisation.

Structure lexicale

HEADER (string)<exp>

L'attribut HEADER est spécifié dans le Tableau 60.

Tableau 60 – Attribut HEADER

Utilisation	Attribut	Description
m	string	Chaîne qui est utilisée pour la communication avec des protocoles de communication spécifiques.

7.7.2.10 MODULE (vide)**7.7.2.11 API****Objet**

L'attribut API (Application Process Identifier) spécifie le profil de l'utilisateur et sert d'information d'adressage supplémentaire. La valeur API par défaut est 0 pour une utilisation illimitée.

Structure lexicale

API (integer, reference, expression)<exp>

Les attributs API sont spécifiés dans le Tableau 61.

Tableau 61 – Attributs API

Utilisa-tion	Attribut	Description
s	integer	Numéro d'API.
s	reference	Référence à une instance de VARIABLE contenant le numéro d'API.
s	expression	Expression qui doit être évaluée à un entier non négatif pour le numéro d'API (pour la description des expressions, voir 7.40).

7.7.2.12 POST_RQSTRECEIVE_ACTIONS

Objet

POST_RQSTRECEIVE_ACTIONS ne doit être pris en charge que par les simulateurs d'appareil.

L'attribut POST_RQSTUPDATE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés par le simulateur d'appareil une fois que toutes les valeurs de la section REQUEST ont été mises à jour et que leur attribut POST_RQSTUPDATE_ACTIONS, le cas échéant, s'exécute, et avant que la section REPLY ne soit générée. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Structure lexicale

POST_RQSTRECEIVE_ACTIONS [(reference)<exp>] +

L'attribut POST_RQSTRECEIVE_ACTIONS est spécifié dans le Tableau 62.

Tableau 62 – Attribut POST_RQSTRECEIVE_ACTIONS

Utilisa-tion	Attribut	Description
m	reference	Référence à un attribut METHOD.

7.8 COMPONENT

7.8.1 Structure générale

Objet

Décrit la configuration du comportement d'un appareil qui est décrit dans l'EDD. Un composant énumère toutes les relations avec les autres composants.

Un composant est constitué d'un fichier DDL, de fichiers d'importation, de fichiers d'inclusion, de dictionnaires, d'images et de fichiers d'aide.

Tous les modules d'un appareil modulaire doivent être décrits avec des attributs COMPONENT. Les relations entre les différents composants sont décrites avec des attributs COMPONENT_RELATIONS. La hiérarchie du catalogue est décrite avec des attributs COMPONENT_FOLDER et COMPONENT. Les expressions conditionnelles peuvent avoir accès à différents composants à l'aide d'OBJECT_REFERENCE. OBJECT_REFERENCE permet d'accéder aux variables et permet d'appeler des méthodes.

Structure lexicale

COMPONENT identifier [LABEL, BYTE_ORDER, CAN_DELETE, CHECK_CONFIGURATION, CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, COMPONENT_RELATIONS, CONNECTION_POINT, DECLARATION, DETECT, EDD, HELP, INITIAL_VALUES, PRODUCT_URI, PROTOCOL, REDUNDANCY, SCAN, SCAN_LIST, SUPPLIED_INTERFACE]

Les attributs sont spécifiés dans le Tableau 63.

Tableau 63 – Attributs COMPONENT

Utilisa-tion	Attribut	Description
m	LABEL	Elément lexical (voir 7.36.9).
o	BYTE_ORDER	Elément lexical (voir 7.8.2.11).
o	CAN_DELETE	Elément lexical (voir 7.8.2.1).
o	CHECK_CONFIGURATION	Elément lexical (voir 7.8.2.2).
o	CLASSIFICATION	Elément lexical (voir 7.36.1).
o	COMPONENT_PARENT	Elément lexical (voir 7.36.2).
o	COMPONENT_PATH	Elément lexical (voir 7.36.3).
o	COMPONENT_RELATIONS	Elément lexical (voir 7.8.2.3).
o	CONNECTION_POINT	Elément lexical (voir 7.8.2.12).
o	DECLARATION	Elément lexical (voir 7.8.2.4).
o	DETECT	Elément lexical (voir 7.8.2.5).
o	EDD	Elément lexical (voir 7.8.2.6).
o	HELP	Elément lexical (voir 7.36.8).
o	INITIAL_VALUES	Elément lexical (voir 7.8.2.7).
o	PRODUCT_URI	Elément lexical (voir 7.8.2.13).
o	PROTOCOL	Elément lexical (voir 7.36.13).
o	REDUNDANCY	Elément lexical (voir 7.8.2.8).
o	SCAN	Elément lexical (voir 7.8.2.9).
o	SCAN_LIST	Elément lexical (voir 7.8.2.10).
o	SUPPLIED_INTERFACE	Elément lexical (voir 7.36.15).

7.8.2 Attributs spécifiques

7.8.2.1 CAN_DELETE

Objet

L'attribut CAN_DELETE est égal à TRUE (vrai) ou FALSE (faux) et indique s'il est sécurisé de supprimer l'instance du module en question d'un appareil modulaire. Cet attribut est utilisé dans des cas d'utilisation en ligne et hors ligne. Par défaut, CAN_DELETE est TRUE. Voici un exemple: les canaux d'un module ne peuvent pas être supprimés; seul le module dans son ensemble peut être supprimé.

Structure lexicale

CAN_DELETE (FALSE, TRUE) <exp>

Les attributs CAN_DELETE sont spécifiés dans le Tableau 64.

Tableau 64 – Attributs CAN_DELETE

Utilisa-tion	Attribut	Description
s	TRUE	Le composant peut être supprimé.
s	FALSE	Le composant ne peut pas être supprimé.

7.8.2.2 CHECK_CONFIGURATION**Objet**

L'attribut CHECK_CONFIGURATION fait référence à une méthode qui doit être appelée par l'application EDD pour confirmer la configuration actuelle de cet attribut COMPONENT. Une valeur de retour BI_SUCCESS indique que la configuration est valide; toutes les autres valeurs indiquent que la configuration est non valide. Si la méthode échoue, la configuration est alors non valide. La méthode référencée ne doit avoir aucun paramètre d'entrée et un paramètre de sortie unique de type DD_STRING qui retourne un message. Ce message doit être affiché à l'utilisateur par l'application EDD. Ce message peut contenir plusieurs explications à une configuration non valide. Dans ce cas, le message peut contenir des sauts de paragraphe et des sauts de ligne. Cette méthode peut être utilisée en ligne et hors ligne.

Structure lexicale

CHECK_CONFIGURATION method_reference

L'attribut CHECK_CONFIGURATION est spécifié dans le Tableau 65.

Tableau 65 – Attribut CHECK_CONFIGURATION

Utilisa-tion	Attribut	Description
m	method_reference	Référence à un attribut METHOD.

7.8.2.3 COMPONENT_RELATIONS**Objet**

L'attribut COMPONENT_RELATIONS répertorie les relations avec d'autres composants que l'appareil modulaire ou module peut prendre en charge. Cette EDD peut répertorier des modules apparentés, des modules enfants et des modules parents.

Structure lexicale

COMPONENT_RELATIONS [(component-relation-reference)<exp>]+

L'attribut COMPONENT_RELATIONS est spécifié dans le Tableau 66.

Tableau 66 – Attribut COMPONENT_RELATIONS

Utilisa-tion	Attribut	Description
m	component-relation-reference	Liste d'identificateurs de relation de composant. Des expressions conditionnelles sont admises.

7.8.2.4 DECLARATION

Objet

L'attribut DECLARATION spécifie des constructions de base associées au composant.

Structure lexicale

DECLARATION [item] +

L'attribut DECLARATION est spécifié dans le Tableau 67.

Tableau 67 – Attribut DECLARATION

Utilisa-tion	Attribut	Description
m	item	Déclarations de constructions de base (voir 7.1.4).

7.8.2.5 DETECT

Objet

L'attribut DETECT indique la méthode qu'il convient que l'application EDD appelle pour détecter un composant.

Dans HART, il convient que l'application EDD lise la commande 75 pour identifier un composant.

Structure lexicale

DETECT method_reference

L'attribut DETECT est spécifié dans le Tableau 68.

Tableau 68 – Attribut DETECT

Utilisa-tion	Attribut	Description
s	method_reference	Référence à une méthode de détection.

7.8.2.6 EDD

Objet

L'attribut EDD spécifie un fichier qui contient le reste de l'EDD. Celui-ci est exigé uniquement si la description de composant se trouve dans un fichier séparé.

Structure lexicale

EDD string

L'attribut EDD est spécifié dans le Tableau 69.

Tableau 69 – Attribut EDD

Utilisa-tion	Attribut	Description
m	string	Chaîne indiquant le fichier qui contient le reste de l'EDD.

7.8.2.7 INITIAL_VALUES**Objet**

L'attribut INITIAL_VALUES est un ensemble de valeurs d'initialisation pour VARIABLE, VALUE_ARRAY ou LIST. Même si les attributs VARIABLE ont un attribut INITIAL_VALUE ou DEFAULT_VALUE, les attributs INITIAL_VALUES de COMPONENT doivent être utilisés pour l'initialisation.

Il convient de les utiliser, par exemple, pour définir différentes initialisations pour différentes utilisations, par exemple la mesure de pression différentielle, de niveau et de débit pour un transducteur de pression.

Structure lexicale

INITIAL_VALUES [reference, [(value<exp>)*, ((value<exp>)+)*]]+

Les attributs INITIAL_VALUES sont spécifiés dans Tableau 65.

Tableau 70 – Attribut INITIAL_VALUES

Utilisa-tion	Attribut	Description
s	reference	Référence à un attribut VARIABLE, VALUE_ARRAY ou LIST.
s	value	Valeur pour initialiser l'attribut VARIABLE référencé. Pour un attribut VALUE_ARRAY ou LIST, un jeu de valeurs. Peut être imbriqué pour un attribut LIST ou VALUE_ARRAY multidimensionnel.

7.8.2.8 REDUNDANCY**Objet**

L'attribut REDUNDANCY contient un entier (par défaut égal à 1) qui indique combien de connecteurs redondants sont pris en charge par l'appareil en question.

Structure lexicale

REDUNDANCY (integer)<exp>

L'attribut REDUNDANCY est spécifié dans le Tableau 71.

Tableau 71 – Attribut REDUNDANCY

Utilisa-tion	Attribut	Description
o	integer	Nombre de points de connexion.

7.8.2.9 SCAN

Objet

L'attribut SCAN indique la méthode qu'il convient que l'application EDD appelle pour créer une liste d'actions d'un appareil modulaire.

Dans HART, il convient que l'application EDD lise avec la commande 74 la liste des modules existants.

Structure lexicale

SCAN method_reference

L'attribut SCAN est spécifié dans le Tableau 72.

Tableau 72 – Attribut SCAN

Utilisa-tion	Attribut	Description
s	method_reference	Référence à la méthode d'analyse.

7.8.2.10 SCAN_LIST

Objet

L'attribut SCAN_LIST est une référence à la liste de résultats de sous-composants qui est créée par la méthode d'analyse.

Dans HART, il convient que l'application EDD lise avec la commande 74 la liste des modules existants.

Structure lexicale

SCAN_LIST reference

L'attribut SCAN_LIST est spécifié dans le Tableau 73.

Tableau 73 – Attribut SCAN_LIST

Utilisa-tion	Attribut	Description
s	reference	Référence à la liste d'analyse.

7.8.2.11 BYTE_ORDER

Objet

L'attribut BYTE_ORDER indique l'ordre des octets utilisé lors de l'envoi de messages de communication entre composants.

Structure lexicale

BYTE_ORDER (BIG_ENDIAN, LITTLE_ENDIAN)

Les attributs BYTE_ORDER sont spécifiés dans le Tableau 74.

Tableau 74 – Attributs BYTE_ORDER

Utilisa-tion	Attribut	Description
s	BIG_ENDIAN	Les octets de poids fort sont transmis avant les octets de poids faible.
s	LITTLE_ENDIAN	Les octets de poids faible sont transmis avant les octets de poids fort.

7.8.2.12 CONNECTION_POINT**Objet**

L'attribut CONNECTION_POINT spécifie une référence à un attribut COLLECTION qui décrit les attributs d'un point de connexion. COLLECTION doit faire référence aux définitions de VARIABLE qui représentent les attributs de point de connexion du modèle d'information spécifié.

Structure lexicale

CONNECTION_POINT reference

L'attribut DEVICE_REVISION est spécifié dans le Tableau 75.

Tableau 75 – Attribut CONNECTION_POINT

Utilisa-tion	Attribut	Description
m	collection_reference	Référence à une instance de COLLECTION.

7.8.2.13 PRODUCT_URI**Objet**

L'attribut PRODUCT_URI est un attribut chaîne. La chaîne est une référence à une URI de produit de serveur de communication qui permet au serveur d'identifier le serveur de communication en fonction du service OPC UA Discovery (voir l'IEC 62541-4). La valeur de l'attribut correspond à l'argument RegisterServer RegisteredServer:serverUri. L'URI du produit doit contenir le nom de l'entreprise et le nom du produit.

Structure lexicale

PRODUCT_URI string

L'attribut PRODUCT_URI est spécifié dans le Tableau 76.

Tableau 76 – Attribut PRODUCT_URI

Utilisa-tion	Attribut	Description
m	string	Littéral de chaîne qui sert de référence à une URI de produit de serveur de communication.

7.9 COMPONENT_FOLDER**Objet**

COMPONENT_FOLDER décrit un dossier dans un catalogue. Les dossiers peuvent comprendre d'autres dossiers. Les dossiers permettent de grouper des composants, par

exemple les composants d'une famille de produits. Les attributs LABEL des dossiers sont associés à COMPONENT_PATH.

Structure lexicale

COMPONENT_FOLDER identifier [LABEL, CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, HELP, PROTOCOL]

Les attributs COMPONENT_FOLDER sont spécifiés dans le Tableau 77.

Tableau 77 – Attributs COMPONENT_FOLDER

Utilisa-tion	Attribut	Description
m	LABEL	Elément lexical (voir 7.36.9).
o	CLASSIFICATION	Elément lexical (voir 7.36.1).
o	COMPONENT_PARENT	Elément lexical (voir 7.36.2).
o	COMPONENT_PATH	Elément lexical (voir 7.36.3).
o	HELP	Elément lexical (voir 7.36.8).
o	PROTOCOL	Elément lexical (voir 7.36.13).

7.10 COMPONENT_REFERENCE

Objet

COMPONENT_REFERENCE désigne un autre composant via l'attribut COMPONENT_PARENT ou avec les attributs PROTOCOL, CLASSIFICATION, MANUFACTURER et COMPONENT_PATH. Si un dossier est référencé, tous les composants qui sont inclus dans ce dossier sont référencés.

Si l'attribut COMPONENT_REFERENCE correspond à un dossier contenant une EDD unique, COMPONENT_REFERENCE concerne un appareil unique. Sinon, une famille d'appareils est indiquée. Sinon, une famille d'appareils est indiquée.

A la place de CLASSIFICATION, un composant peut également être référencé avec les attributs DEVICE_TYPE et DEVICE_REVISION.

Structure lexicale

COMPONENT_REFERENCE identifier [CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, DEVICE_REVISION, DEVICE_TYPE, MANUFACTURER, PROTOCOL]

Les attributs COMPONENT_REFERENCE sont spécifiés dans le Tableau 78.

Tableau 78 – Attributs COMPONENT_REFERENCE

Utilisa-tion	Attribut	Description
o	CLASSIFICATION	Elément lexical (voir 7.36.1).
o	COMPONENT_PARENT	Elément lexical (voir 7.36.2).
o	COMPONENT_PATH	Elément lexical (voir 7.36.3).
o	DEVICE_REVISION	Elément lexical (voir 7.2.2.2).
o	DEVICE_TYPE	Elément lexical (voir 7.2.2.3).
o	MANUFACTURER	Elément lexical (voir 7.2.2.6), par défaut à l'attribut MANUFACTURER de l'EDD.
o	PROTOCOL	Elément lexical (voir 7.36.13).

7.11 COMPONENT_RELATION

7.11.1 Structure générale

Objet

COMPONENT_RELATION spécifie une relation entre un module et un ou plusieurs autres modules et les contraintes qui doivent être remplies pour instancier ces autres modules.

Structure lexicale

COMPONENT_RELATION identifier [COMPONENTS, LABEL, RELATION_TYPE, ADDRESSING, HELP, MAXIMUM_NUMBER, MINIMUM_NUMBER, REQUIRED_INTERFACE, SUPPLIED_INTERFACE]

Les attributs COMPONENT_RELATION sont spécifiés dans le Tableau 79.

Tableau 79 – Attributs COMPONENT_RELATION

Utilis-ation	Attribut	Description
m	COMPONENTS	Elément lexical (voir 7.11.2.1).
m	LABEL	Elément lexical (voir 7.36.9).
m	RELATION_TYPE	Elément lexical (voir 7.11.2.2).
o	ADDRESSING	Elément lexical (voir 7.11.2.3).
o	HELP	Elément lexical (voir 7.36.8).
o	MAXIMUM_NUMBER	Elément lexical (voir 7.11.2.4).
o	MINIMUM_NUMBER	Elément lexical (voir 7.11.2.5).
o	REQUIRED_INTERFACE	Elément lexical (voir 7.11.2.6).
o	SUPPLIED_INTERFACE	Elément lexical (voir 7.36.15).

7.11.2 Attributs spécifiques

7.11.2.1 COMPONENTS

Objet

L'attribut COMPONENTS répertorie les modules ou les appareils modulaires admis et les contraintes liées à leur utilisation.

Structure lexicale

COMPONENTS [component-reference, AUTO_CREATE, MINIMUM_NUMBER, MAXIMUM_NUMBER, FILTER, REQUIRED_RANGES]<exp>+

Les attributs COMPONENTS sont spécifiés dans le Tableau 80.

Tableau 80 – Attributs COMPONENTS

Utilisation	Attribut	Description
m	component-reference	COMPONENT_FOLDER ou COMPONENT dans la relation.
o	AUTO_CREATE	Nombre de composants automatiquement créés. Structure lexicale AUTO_CREATE (expression)<exp>
o	FILTER	Utilisé si la référence désigne un attribut COMPONENT_FOLDER et spécifie une expression booléenne qui permet de filtrer les composants. Des composants sont inclus si l'expression est TRUE. Pour filtrer des attributs COMPONENT spéciaux, des références au type de composant sont utilisées dans l'expression. La condition est évaluée pour chaque type contenu dans l'attribut COMPONENT_FOLDER. Structure lexicale FILTER (TRUE, FALSE)<exp>
o	MAXIMUM_NUMBER	Nombre maximal admis de ce composant. Structure lexicale MAXIMUM_NUMBER (expression)<exp>
o	MINIMUM_NUMBER	Nombre minimal exigé de ce composant. Structure lexicale MINIMUM_NUMBER (expression)<exp>
o	REQUIRED_RANGES	Variables avec intervalles. Dans une configuration valide, les variables doivent être dans l'intervalle spécifié. Il est typiquement utilisé pour définir des restrictions pour l'adressage (par exemple, une plage). Structure lexicale REQUIRED_RANGES [reference, [MAX_VALUE, m]*, [MIN_VALUE, n]*]+

7.11.2.2 RELATION_TYPE

Objet

L'attribut RELATION_TYPE définit la relation entre le composant en question et d'autres composants (enfants, parents ou apparentés).

Structure lexicale

RELATION_TYPE [CHILD_COMPONENT, PARENT_COMPONENT, SIBLING_COMPONENT]

Les attributs RELATION_TYPE sont spécifiés dans le Tableau 81.

Tableau 81 – Attributs RELATION_TYPE

Utilisa-tion	Attribut	Description
s	CHILD_COMPONENT	Liste de composants qui sont des sous-modules admis.
s	PARENT_COMPONENT	Liste de composants qui sont des modules ou des appareils modulaires parents admis pour ce composant.
s	SIBLING_COMPONENT	Relation entre des composants apparentés qui peut exiger ou exclure d'autres composants. Il peut s'agir d'exigence lorsque MINIMUM_NUMBER > 0. Il peut s'agir d'exclusion lorsque MAXIMUM_NUMBER = 0.
s	NEXT_COMPONENT	Relation de parenté spéciale avec le composant suivant relative à l'attribut ADDRESSING. NEXT_COMPONENT peut uniquement être utilisé si ADDRESSING est spécifié. Non applicable pour HART.
s	PREV_COMPONENT	Relation de parenté spéciale avec le composant précédent relative à l'attribut ADDRESSING. PREV_COMPONENT peut uniquement être utilisé si ADDRESSING est spécifié. Non applicable pour HART.

7.11.2.3 ADDRESSING**Objet**

L'attribut ADDRESSING spécifie une liste de variables qui adresse un composant de façon unique.

Structure lexicale

ADDRESSING variable-reference+

L'attribut ADDRESSING est spécifié dans le Tableau 82.

Tableau 82 – Attribut ADDRESSING

Utilisa-tion	Attribut	Description
m	variable-reference	Référence à une variable.

7.11.2.4 MAXIMUM_NUMBER**Objet**

L'attribut MAXIMUM_NUMBER spécifie le nombre de composants qui sont admis. La valeur par défaut de MAXIMUM_NUMBER est 1.

Structure lexicale

MAXIMUM_NUMBER (integer) <exp>

L'attribut MAXIMUM_NUMBER est spécifié dans le Tableau 83.

Tableau 83 – Attribut MAXIMUM_NUMBER

Utilisa-tion	Attribut	Description
o	integer	Nombre total maximal admis de l'ensemble des composants répertoriés.

7.11.2.5 MINIMUM_NUMBER**Objet**

L'attribut MINIMUM_NUMBER spécifie le nombre de composants qui sont exigés. La valeur par défaut de MINIMUM_NUMBER est 0.

Structure lexicale

MINIMUM_NUMBER (integer) <exp>

L'attribut MINIMUM_NUMBER est spécifié dans le Tableau 84.

Tableau 84 – Attribut MINIMUM_NUMBER

Utilisa-tion	Attribut	Description
o	integer	Nombre total minimal exigé de l'ensemble des composants répertoriés.

7.11.2.6 REQUIRED_INTERFACE**Objet**

L'attribut REQUIRED_INTERFACE répertorie les interfaces exigées du composant associé.

Structure lexicale

REQUIRED_INTERFACE interface-reference+

L'attribut REQUIRED_INTERFACE est spécifié dans le Tableau 85.

Tableau 85 – Attribut REQUIRED_INTERFACE

Utilisa-tion	Attribut	Description
m	interface-reference	Référence à une interface.

7.12 CONNECTION (vide)**7.13 DOMAIN (vide)****7.14 EDIT_DISPLAY****7.14.1 Structure générale****Objet**

Un attribut EDIT_DISPLAY définit comment les données seront gérées pour l'affichage et l'édition. Il est utilisé pour grouper des éléments conjointement pour édition.

Cette construction est incluse pour la rétrocompatibilité. Pour de futures implémentations, il convient d'utiliser la construction de base plus générale de MENU.

Structure lexicale

```
EDIT_DISPLAY  identifier  [EDIT_ITEMS,      LABEL,      DISPLAY_ITEMS,
PRE_EDIT_ACTIONS, POST_EDIT_ACTIONS]
```

Les attributs EDIT_DISPLAY sont spécifiés dans le Tableau 86.

Tableau 86 – Attributs EDIT_DISPLAY

Utilisa-tion	Attribut	Description
m	EDIT_ITEMS	Elément lexical (voir 7.14.2.1).
m	LABEL	Elément lexical (voir 7.36.9).
o	DISPLAY_ITEMS	Elément lexical (voir 7.14.2.2).
o	POST_EDIT_ACTIONS	Elément lexical (voir 7.14.2.3).
o	PRE_EDIT_ACTIONS	Elément lexical (voir 7.14.2.4).

7.14.2 Attributs spécifiques

7.14.2.1 EDIT_ITEMS

Objet

L'attribut EDIT_ITEMS définit l'ensemble des éléments de données qui doivent être présentés à l'utilisateur, et peuvent être édités par l'utilisateur: VARIABLES, relations WRITE_AS_ONE, BLOCK_A PARAMETERS et éléments de BLOCK_A PARAMETERS (c'est-à-dire, respectivement des champs ou éléments de BLOCK_A PARAMETERS de type RECORD ou VALUE_ARRAY).

NOTE Les détails spécifiques pour la présentation et l'édition d'une donnée (par exemple, valeur par défaut, facteur d'échelle, intervalle) sont spécifiés dans les définitions de chaque élément individuel référencé.

Si une relation WRITE_AS_ONE apparaît en tant qu'attribut EDIT_ITEM, il convient que l'utilisateur examine toutes les variables dans la relation WRITE_AS_ONE. L'utilisateur peut ne pas modifier toutes les valeurs mais il convient au moins qu'il valide que les valeurs actuelles sont acceptables.

Structure lexicale

```
EDIT_ITEMS [ (reference) <exp> ] +
```

L'attribut EDIT_ITEMS est spécifié dans le Tableau 87.

Tableau 87 – Attribut EDIT_ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de VARIABLE, WRITE_AS_ONE, BLOCK_A PARAMETERS ou à un élément de BLOCK_A PARAMETERS. Les références admises sont spécifiques au profil.

7.14.2.2 DISPLAY_ITEMS

Objet

L'attribut DISPLAY_ITEMS définit l'ensemble des éléments de données qui doivent être présentés à l'utilisateur pour information, c'est-à-dire que ceux-ci ne doivent pas être édités: VARIABLES, BLOCK_A PARAMETERS et éléments de BLOCK_A PARAMETERS (c'est-à-dire, respectivement des champs ou éléments de BLOCK_A PARAMETERS de type RECORD ou VALUE_ARRAY).

NOTE Les détails spécifiques pour la présentation d'une donnée (par exemple, valeur par défaut, facteur d'échelle, intervalle) sont définis dans les définitions de chaque élément individuel référencé.

Structure lexicale

DISPLAY_ITEMS [(reference)<exp>]+

L'attribut DISPLAY_ITEMS est spécifié dans le Tableau 88.

Tableau 88 – Attribut DISPLAY_ITEM

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de VARIABLE, WRITE_AS_ONE, BLOCK_A PARAMETERS ou à un élément de BLOCK_A PARAMETERS. Les références admises sont spécifiques au profil.

7.14.2.3 POST_EDIT_ACTIONS

Objet

L'attribut POST_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après que l'utilisateur a fini le traitement de l'objet EDIT_DISPLAY. Si EDIT_DISPLAY est abandonné, les attributs POST_EDIT_ACTIONS ne sont pas exécutés. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine anormalement, les attributs METHOD suivants ne sont pas exécutés.

Les attributs POST_EDIT_ACTIONS éventuels d'une entité référencée dans les attributs EDIT_ITEMS doivent être exécutés lorsqu'une nouvelle valeur a été entrée. Les attributs POST_EDIT_ACTIONS d'EDIT_DISPLAY sont exécutés uniquement après que tous ces attributs POST_EDIT_ACTIONS individuels sont terminés.

L'attribut POST_EDIT_ACTIONS ne doit pas être exprimé en utilisant des instructions conditionnelles IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

POST_EDIT_ACTIONS [reference]+, DEFINITION

Les attributs POST_EDIT_ACTIONS sont spécifiés dans le Tableau 89.

7.14.2.4 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement lorsqu'EDIT_DISPLAY est activé. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine

anormalement, les attributs METHOD suivants ne sont pas exécutés et l'activation d'EDIT_DISPLAY est annulée.

Les attributs PRE_EDIT_ACTIONS d'EDIT_DISPLAY doivent être exécutés avant tout attribut PRE_EDIT_ACTIONS défini des entités référencées.

L'attribut PRE_EDIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

NOTE Les attributs PRE_EDIT_ACTIONS de tous les objets VARIABLE associés sont exécutés immédiatement avant que l'utilisateur ne commence effectivement à les éditer, par exemple, en plaçant un curseur dans un champ d'édition. L'affichage d'objets VARIABLE sans édition ne déclenche pas les attributs PRE_EDIT_ACTIONS des objets VARIABLE.

Structure lexicale

PRE_EDIT_ACTIONS [reference]+, DEFINITION

Les attributs PRE_EDIT_ACTIONS sont spécifiés dans le Tableau 89.

Tableau 89 – Attributs POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de METHOD ou à une instance de METHOD avec arguments.
o	DEFINITION	Elément lexical (voir 7.36.4).

7.15 FILE

7.15.1 Structure générale

Objet

FILE décrit une zone de stockage persistant qui est gérée par l'application EDD. La zone mémoire doit contenir uniquement des données. L'attribut MEMBERS définit la structure de la zone mémoire, qui doit être fixe, ce qui signifie que l'attribut MEMBERS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT.

Structure lexicale

FILE identifier [MEMBERS, HANDLING, HELP, IDENTITY, LABEL, SHARED, ON_UPDATE_ACTIONS]

Les attributs FILE sont spécifiés dans le Tableau 90.

Tableau 90 – Attributs FILE

Utilisa-tion	Attribut	Description
m	MEMBERS	Elément lexical (voir 7.36.12).
o	HANDLING	Elément lexical (voir 7.36.6).
o	HELP	Elément lexical (voir 7.36.8).
o	IDENTITY	Elément lexical (voir 7.36.21). Spécifie le nom du fichier associé, par exemple un référentiel de données d'ingénierie précalculées (par exemple, géométries de tank ou tableaux de flux).
o	LABEL	Elément lexical (voir 7.36.9).
o	SHARED	Elément lexical (voir 7.15.2.1).
o	ON_UPDATE_ACTIONS	Elément lexical (voir 7.15.2.2).

7.15.2 Attributs spécifiques**7.15.2.1 SHARED****Objet**

L'attribut SHARED spécifie si le fichier est partagé par toutes les instances du type d'appareil. Si SHARED n'est pas spécifié ou si sa valeur est FALSE, le fichier est associé à l'instance d'appareil spécifique.

Structure lexicale

SHARED (TRUE, FALSE)

Les attributs SHARED sont spécifiés dans le Tableau 91.

Tableau 91 – Attributs SHARED

Utilisa-tion	Attribut	Description
s	TRUE	Le fichier est partagé dans toutes les instances du même type d'appareil.
s	FALSE	Le fichier est spécifique à l'instance d'appareil. La valeur par défaut de SHARED est FALSE.

7.15.2.2 ON_UPDATE_ACTIONS**Objet**

ON_UPDATE_ACTIONS ne doit être pris en charge que par les simulateurs d'appareil.

L'attribut ON_UPDATE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés si le fichier partagé est remplacé ou mis à jour par une application étrangère externe. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Structure lexicale

ON_UPDATE_ACTIONS [(reference) <exp>] +

L'attribut ON_UPDATE_ACTIONS est spécifié dans le Tableau 92.

Tableau 92 – Attribut ON_UPDATE_ACTIONS

Utilisa-tion	Attribut	Description
m	reference	Référence à un attribut METHOD.

7.16 GRAPH

7.16.1 Structure générale

Objet

GRAPH décrit un graphique utilisé pour afficher des données provenant d'un appareil de l'application EDD. Un attribut GRAPH est utilisé pour afficher un ensemble de données qui est stocké dans un appareil, par opposition à un attribut CHART qui est utilisé pour afficher des valeurs de données collectées en continu depuis un appareil.

Structure lexicale

GRAPH identifier [MEMBERS, CYCLE_TIME, HEIGHT, HELP, LABEL, VALIDITY, VISIBILITY, WIDTH, X_AXIS]

Les attributs GRAPH sont spécifiés dans le Tableau 93.

Tableau 93 – Attributs GRAPH

Utilisa-tion	Attribut	Description
m	MEMBERS	Elément lexical (voir 7.36.12).
o	CYCLE_TIME	Elément lexical (voir 7.16.2.1).
o	HEIGHT	Elément lexical (voir 7.36.7).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	WIDTH	Elément lexical (voir 7.36.17).
o	X_AXIS	Elément lexical (voir 7.16.2.2).

7.16.2 Attributs spécifiques

7.16.2.1 CYCLE_TIME

Objet

L'attribut CYCLE_TIME spécifie le temps, en ms, durant lequel l'application EDD doit attendre entre des lectures successives des valeurs sources de l'appareil. Par défaut, un attribut GRAPH n'a pas de durée de cycle.

Structure lexicale

CYCLE_TIME (integer)<exp>

L'attribut CYCLE_TIME est spécifié dans le Tableau 94.

Tableau 94 – Attribut CYCLE_TIME

Utilisa-tion	Attribut	Description
m	integer	Temps, en ms, entre des lectures successives des valeurs sources de l'appareil. Après que GRAPH est affiché par une application EDD, CYCLE_TIME ne doit pas changer.

7.16.2.2 X_AXIS**Objet**

L'attribut X_AXIS spécifie la façon dont X_AXIS s'affiche. Chaque attribut WAVEFORM sur un attribut GRAPH partage un attribut X_AXIS commun, mais chaque attribut WAVEFORM sur un attribut GRAPH a son propre attribut Y_AXIS. Par défaut, l'affichage de l'axe X d'un attribut GRAPH sans attribut X_AXIS est dérivé des attributs WAVEFORM sur l'attribut GRAPH.

Structure lexicale

X_AXIS (reference) <exp>

L'attribut X_AXIS est spécifié dans le Tableau 95.

Tableau 95 – Attribut X_AXIS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance d'AXIS.

7.17 GRID**7.17.1 Structure générale****Objet**

L'attribut GRID décrit une matrice de données provenant d'un appareil qui sont destinées à être affichées par l'application EDD. Un attribut GRID est utilisé pour afficher des vecteurs de données avec l'en-tête ou la description des données dans ce vecteur. Les vecteurs sont affichés horizontalement (lignes) ou verticalement (colonnes), comme spécifié par l'attribut ORIENTATION.

Structure lexicale

GRID identifier [VECTORS, HANDLING, HEIGHT, HELP, LABEL, ORIENTATION, VALIDITY, VISIBILITY, WIDTH]

Les attributs GRID sont spécifiés dans le Tableau 96.

Tableau 96 – Attributs GRID

Utilisa-tion	Attribut	Description
m	VECTORS	Elément lexical (voir 7.17.2.1).
o	HANDLING	Elément lexical (voir 7.36.6).
o	HEIGHT	Elément lexical (voir 7.36.7).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	ORIENTATION	Elément lexical (voir 7.17.2.2).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	WIDTH	Elément lexical (voir 7.36.17).

7.17.2 Attributs spécifiques

7.17.2.1 VECTORS

Objet

L'attribut VECTORS spécifie le contenu de l'élément GRID. Chaque vecteur est affiché dans l'ordre. Pour un attribut GRID vertical, les vecteurs sont affichés de gauche à droite. Pour un attribut GRID horizontal, les vecteurs sont affichés de haut en bas. Chaque vecteur est constitué d'une description qui constitue une étiquette pour le vecteur suivi par les données à afficher.

Structure lexicale

VECTORS [description, (reference, string, double, float, integer, unsigned_integer) <exp>+]

Les attributs VECTORS sont spécifiés dans le Tableau 97.

Tableau 97 – Attributs VECTORS

Utilisa-tion	Attribut	Description
m	description	Chaîne contenant une brève description des données contenues dans le vecteur.
o	reference	Référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY, un élément d'une instance de REFERENCE_ARRAY, une instance de VALUE_ARRAY ou une instance de REFERENCE_ARRAY.
	string	Chaîne constante ou référence représentée par une chaîne.
o	double	Constante à virgule flottante à double précision.
o	float	Constante à virgule flottante à simple précision.
o	integer	Constante entière.
o	unsigned_integer	Constante entière non signée.

7.17.2.2 ORIENTATION

Objet

L'attribut ORIENTATION spécifie la direction dans laquelle les vecteurs sont affichés. Par défaut, un attribut GRID sans attribut ORIENTATION est affiché verticalement.

Structure lexicale

ORIENTATION (HORIZONTAL, VERTICAL)<exp>

Les attributs ORIENTATION sont spécifiés dans le Tableau 98.

Tableau 98 – Attributs ORIENTATION

Utilisa-tion	Attribut	Description
s	HORIZONTAL	Les vecteurs sont affichés en lignes.
s	VERTICAL	Les vecteurs sont affichés en colonnes.

7.18 IMAGE

7.18.1 Structure générale

Objet

IMAGE décrit une image visuelle.

Structure lexicale

IMAGE identifier [PATH, HELP, LABEL, LINK, VALIDITY, VISIBILITY]

Les attributs IMAGE sont spécifiés dans le Tableau 99.

Tableau 99 – Attributs IMAGE

Utilisa-tion	Attribut	Description
m	PATH	Elément lexical (voir 7.18.2.1).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	LINK	Elément lexical (voir 7.18.2.2).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).

7.18.2 Attributs spécifiques

7.18.2.1 PATH

Objet

L'attribut PATH spécifie le nom du fichier de l'image. Des images localisées multiples peuvent être spécifiées en utilisant la même technique de localisation que celle utilisée avec les littéraux de chaîne. Le code pays zz peut être utilisé pour spécifier une version basse résolution de l'image, ceci est utile pour les applications EDD qui fonctionnent sur des appareils mobiles ayant une mémoire restreinte.

EXEMPLE " |en|english.jpeg|de|german.jpeg|en zz|english-lowres.jpeg|de zz|german-lowres.jpeg"

Structure lexicale

PATH (file-name)<exp>

L'attribut PATH est spécifié dans le Tableau 100.

Tableau 100 – Attribut PATH

Utilisa-tion	Attribut	Description
m	file-name	Littéral de chaîne qui identifie le nom de fichier de l'image.

7.18.2.2 LINK

Objet

L'attribut LINK spécifie un attribut MENU ou METHOD qui peut être accessible depuis IMAGE. Cela permet d'obtenir des informations additionnelles concernant l'attribut IMAGE.

Structure lexicale

LINK (reference)<exp>

L'attribut LINK est spécifié dans le Tableau 101.

Tableau 101 – Attribut LINK

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de MENU ou à une instance de METHOD.

7.19 IMPORT

7.19.1 Structure générale

Objet

Les constructions de langage EDDL sont suffisantes pour décrire un appareil individuel quelconque. Cependant, des mécanismes additionnels doivent décrire des révisions multiples d'un appareil individuel ou décrire des parties d'un appareil générique. Pour assurer ces mécanismes, une EDD peut importer une autre EDD. L'EDD importée peut elle-même importer également d'autres EDD.

Avec ces mécanismes, l'EDD d'une nouvelle version de l'appareil peut être spécifiée en important simplement l'EDD de l'ancienne version de l'appareil et en spécifiant les modifications d'éléments.

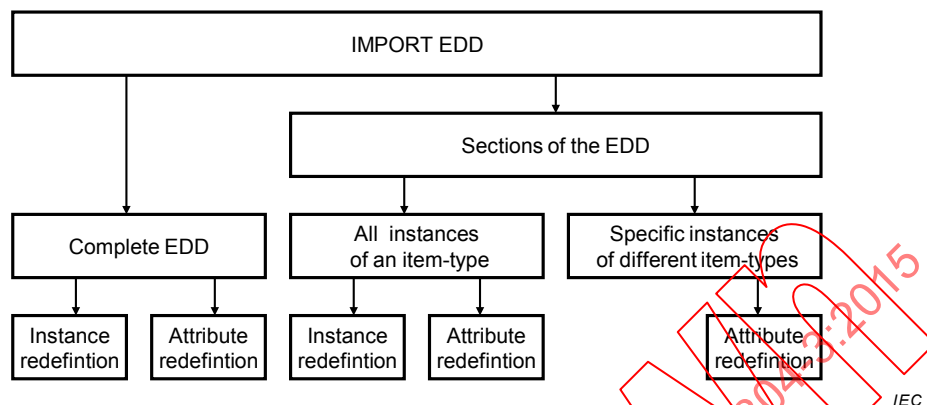
NOTE Ce type d'EDD est appelé description delta, car l'EDD entière est spécifiée par des modifications d'une EDD existante.

Trois types d'importation doivent être pris en charge:

- importation de tous les éléments, où tous les éléments de l'EDD externe sont importés;
- importation des éléments d'un type spécifié, où uniquement les éléments des types spécifiés sont importés. Si un type spécifié est importé, des éléments spécifiques du même type ne peuvent pas être également importés;

- importation d'un élément spécifique.

L'identificateur d'un élément EDD importé ne doit pas être modifié. La Figure 29 montre les mécanismes d'importation EDDL.



Anglais	Français
IMPORT EDD	IMPORTATION EDD
Sections of the EDD	Sections de l'EDD
Complete EDD	EDD entière
All instances of an item-type	Toutes les instances d'un attribut item-type
Specific instances of different item-types	Instances spécifiques de différents attributs item-type
Instance redefinition	Redéfinition de l'instance
Attribute redefinition	Redéfinition de l'attribut

Figure 29 – Mécanismes d'importation EDDL

Une autre méthode d'importation consiste à référencer l'EDD par le nom du fichier.

Structure lexicale

```

IMPORT ((DD_REVISION, DEVICE_REVISION, DEVICE_TYPE, EDD_PROFILE,
EDD_VERSION, MANUFACTURER, MANUFACTURER_EXT), FILE_NAME), (EVERYTHING,
(Axes, BLOBS, BLOCKS_A, BLOCKS_B, CHART, COLLECTIONS, COMMANDS,
COMPONENTS, COMPONENT_FOLDERS, COMPONENT_REFERENCES,
COMPONENT_RELATIONS, EDIT_DISPLAYS, FILES, GRAPHS, GRIDS, IMAGES,
INTERFACES, LISTS, MENUS, METHODS, PLUGINS, RECORDS, REFERENCE_ARRAYS,
REFRESHES, RELATIONS, RESPONSE_CODES, SOURCES, TEMPLATES, UNITS,
VALUE_ARRAYS, VARIABLES, VARIABLE_LISTS, WAVEFORMS, WRITE_AS_ONES,
reference*,
[DELETE identifiant]*,
[REDEFINE identifiant]*,
attribute-redefinition*))
  
```

Les attributs IMPORT sont spécifiés dans le Tableau 102.

Tableau 102 – Importation de description d'un appareil

Utilisation	Attribut	Description
o	attribute-redefinition	Voir 7.19.2.
o	AXES	Toutes les instances d'AXIS d'une EDD externe sont importées.
o	BLOBS	Toutes les instances de BLOB d'une EDD externe sont importées.
o	BLOCKS_A	Toutes les instances de BLOCK_A d'une EDD externe sont importées.
o	BLOCKS_B	Toutes les instances de BLOCK_B d'une EDD externe sont importées.
o	CHARTS	Toutes les instances de CHART d'une EDD externe sont importées.
o	COLLECTIONS	Toutes les instances de COLLECTION d'une EDD externe sont importées.
o	COMMANDS	Toutes les instances de COMMAND d'une EDD externe sont importées.
o	COMPONENTS	Toutes les instances de COMPONENT d'une EDD externe sont importées.
o	COMPONENT_FOLDERS	Toutes les instances de COMPONENT_FOLDER d'une EDD externe sont importées.
o	COMPONENT_REFERENCES	Toutes les instances de COMPONENT_REFERENCE d'une EDD externe sont importées.
o	COMPONENT_RELATIONS	Toutes les instances de COMPONENT_RELATION d'une EDD externe sont importées.
m	DD_REVISION	Voir 7.2.2.1.
o	DELETE	Une instance importée doit être supprimée.
m	DEVICE_REVISION	Voir 7.2.2.2.
m	DEVICE_TYPE	Voir 7.2.2.3.
o	EDD_PROFILE	Voir 7.2.2.4.
o	EDD_VERSION	Voir 7.2.2.5.
o	EDIT_DISPLAYS	Toutes les instances d'EDIT_DISPLAY d'une EDD externe sont importées.
o	EVERYTHING	Tous les éléments d'une EDD externe sont importés. Si cette forme est utilisée, aucun type d'élément ni aucune instance individuelle ne peuvent être sélectionnés. Les attributs DELETE, REDEFINE et attribute-redefinition peuvent être utilisés.
o	FILE_NAME	Chaîne qui contient un nom de fichier et son chemin d'accès dans une structure de répertoire. Le fichier contient l'EDD qui doit être importée. Avec ce type de référencement, la spécification des informations d'identification d'EDD n'est pas nécessaire.
o	FILES	Toutes les instances de FILE d'une EDD externe sont importées.
o	GRAPHS	Toutes les instances de GRAPH d'une EDD externe sont importées.
o	GRIDS	Toutes les instances de GRID d'une EDD externe sont importées.
o	IMAGES	Toutes les instances d'IMAGE d'une EDD externe sont importées.
o	INTERFACES	Toutes les instances d'INTERFACE d'une EDD externe sont importées.
o	LISTS	Toutes les instances de LIST d'une EDD externe sont importées.
m	MANUFACTURER	Voir 7.2.2.6.
c	MANUFACTURER_EXT	Voir 7.2.2.7.
o	MENUS	Toutes les instances de MENU d'une EDD externe sont importées.
o	METHODS	Toutes les instances de METHOD d'une EDD externe sont importées.
o	PLUGINS	Toutes les instances de PLUGIN d'une EDD externe sont importées.
o	RECORDS	Toutes les instances de RECORD d'une EDD externe sont importées.
o	REDEFINE	Tous les attributs d'une instance importée doivent être redéfinis.
o	reference	Une instance unique est importée. Une instance unique ne doit pas être importée si elle est déjà incluse dans une importation de type d'élément.
o	REFERENCE_ARRAYS	Toutes les instances de REFERENCE_ARRAY d'une EDD externe sont importées.
o	REFRESHES	Toutes les instances de REFRESH d'une EDD externe sont importées.
o	RELATIONS	Toutes les instances de REFRESH, UNIT et WRITE_AS_ONE d'une EDD externe sont importées.

Utilisa- tion	Attribut	Description
o	RESPONSE_CODES	Toutes les instances de RESPONSE_CODES d'une EDD externe sont importées.
o	SOURCES	Toutes les instances de SOURCE d'une EDD externe sont importées.
o	TEMPLATES	Toutes les instances de TEMPLATE d'une EDD externe sont importées.
o	UNITS	Toutes les instances d'UNIT d'une EDD externe sont importées.
o	VALUE_ARRAYS	Toutes les instances de VALUE_ARRAY d'une EDD externe sont importées.
o	VARIABLE_LISTS	Toutes les instances de VARIABLE_LIST d'une EDD externe sont importées.
o	VARIABLES	Toutes les instances de VARIABLE d'une EDD externe sont importées.
o	WAVEFORMS	Toutes les instances de WAVEFORM d'une EDD externe sont importées.
o	WRITE_AS_ONES	Toutes les instances de WRITE_AS_ONE d'une EDD externe sont importées.

7.19.2 Redéfinitions

7.19.2.1 Généralités

Objet

Des attributs d'une instance importée peuvent être ajoutés, supprimés ou redéfinis. L'ajout, la suppression et la redéfinition n'affectent pas tous les attributs d'une construction de l'EDD de l'élément de base.

Les attributs de redéfinition sont spécifiés dans le Tableau 103.

Tableau 103 – Attributs de redéfinition

Utilisa- tion	Attribut	Description
s	ADD	Ajoute l'attribut spécifié suivant. "A" indique qu'ADD peut être appliqué à l'attribut dans les tableaux des règles de redéfinition (voir du Tableau 104 au Tableau 133).
s	DELETE	Supprime l'attribut spécifié suivant. "D" indique que DELETE peut être appliqué à l'attribut dans les tableaux des règles de redéfinition (voir du Tableau 104 au Tableau 133).
s	REDEFINE	Redéfinit l'attribut spécifié suivant. "R" indique que REDEFINE peut être appliqué à l'attribut dans les tableaux des règles de redéfinition (voir du Tableau 104 au Tableau 133).

7.19.2.2 AXIS

Structure lexicale

AXIS *identifieur*, [[DELETE, REDEFINE], [CONSTANT_UNIT, HELP, LABEL, MAX_VALUE, MIN_VALUE, SCALING]]+

Les règles de redéfinition pour les attributs AXIS sont spécifiées dans le Tableau 103 et le Tableau 104.

Tableau 104 – Règles de redéfinition pour les attributs AXIS

A	D	R	Attribut	Description
	•	•	CONSTANT_UNIT	Elément lexical (voir 7.3.2.1).
	•	•	HELP	Elément lexical (voir 7.3.6.8).
	•	•	LABEL	Elément lexical (voir 7.3.6.9).
	•	•	MAX_VALUE	Elément lexical (voir 7.3.2.2).
	•	•	MIN_VALUE	Elément lexical (voir 7.3.2.3).
	•	•	SCALING	Elément lexical (voir 7.3.2.4).

7.19.2.3 BLOB**Structure lexicale**

BLOB identifier, [[DELETE, REDEFINE], [HANDLING, HELP, IDENTITY, LABEL]]+

Les règles de redéfinition pour les attributs BLOB sont spécifiées dans le Tableau 103 et le Tableau 105.

Tableau 105 – Règles de redéfinition pour les attributs BLOB

A	D	R	Attribut	Description
		•	HANDLING	Elément lexical (voir 7.3.6.6).
	•	•	HELP	Elément lexical (voir 7.3.6.8).
	•	•	IDENTITY	Elément lexical (voir 7.3.6.21).
	•	•	LABEL	Elément lexical (voir 7.3.6.9).

7.19.2.4 BLOCK_A**Structure lexicale**

BLOCK identifier, [[ADD, DELETE, REDEFINE], [CHARACTERISTICS, LABEL, PARAMETERS, parameters-list-element, AXIS_ITEMS, CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISPLAY_ITEMS, FILE_ITEMS, GRAPH_ITEMS, GRID_ITEMS, HELP, IMAGE_ITEMS, LIST_ITEMS, LOCAL_PARAMETERS, local-parameters-list-element, MENU_ITEMS, METHOD_ITEMS, PARAMETER_LISTS, parameter-lists-list-element, PLUGIN_ITEMS, REFERENCE_ARRAY_ITEMS, REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS, WRITE_AS_ONE_ITEMS, CHARTS, charts-element, FILES, files-element, LISTS, lists-element, GRAPHS, graphs-element, GRIDS, grids-element, MENUS, menus-element, METHODS, methods-element, PLUGINS, plugins-element]]+

Les règles de redéfinition pour les attributs BLOCK_A sont spécifiées dans le Tableau 103 et le Tableau 106.

Tableau 106 – Règles de redéfinition pour les attributs BLOCK_A

A	D	R	Attribut	Description
		•	CHARACTERISTICS	Elément lexical (voir 7.4.1.2.1).
		•	LABEL	Elément lexical (voir 7.36.9).
		•	PARAMETERS	Elément lexical (voir 7.4.1.2.2).
•		•	parameters-list-element	La liste des éléments individuels de PARAMETERS peut être modifiée.
	•	•	AXIS_ITEMS	Elément lexical (voir 7.4.1.2.3).
	•	•	CHART_ITEMS	Elément lexical (voir 7.4.1.2.4).
	•	•	COLLECTION_ITEMS	Elément lexical (voir 7.4.1.2.5).
	•	•	EDIT_DISPLAY_ITEMS	Elément lexical (voir 7.4.1.2.6).
	•	•	FILE_ITEMS	Elément lexical (voir 7.4.1.2.7).
	•	•	GRAPH_ITEMS	Elément lexical (voir 7.4.1.2.8).
	•	•	GRID_ITEMS	Eléments lexicaux (voir 7.4.1.2.9).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	IMAGE_ITEMS	Eléments lexicaux (voir 7.4.1.2.10).
	•	•	LIST_ITEMS	Elément lexical (voir 7.4.1.2.11).
		•	LOCAL_PARAMETERS	Elément lexical (voir 7.4.1.2.12).
•		•	local-parameters-list-element	La liste des éléments individuels de LOCAL_PARAMETERS peut être modifiée.
	•	•	MENU_ITEMS	Elément lexical (voir 7.4.1.2.13).
	•	•	METHOD_ITEMS	Elément lexical (voir 7.4.1.2.14).
		•	PARAMETER_LISTS	Elément lexical (voir 7.4.1.2.15).
•		•	parameter-lists-list-element	La liste des éléments individuels de PARAMETERS_LISTS peut être modifiée.
	•	•	PLUGIN_ITEMS	Elément lexical (voir 7.4.1.2.29).
	•	•	REFERENCE_ARRAY_ITEMS	Elément lexical (voir 7.4.1.2.16).
	•	•	REFRESH_ITEMS	Elément lexical (voir 7.4.1.2.17).
	•	•	SOURCE_ITEMS	Elément lexical (voir 7.4.1.2.18).
	•	•	UNIT_ITEMS	Elément lexical (voir 7.4.1.2.19).
	•	•	WAVEFORM_ITEMS	Elément lexical (voir 7.4.1.2.20).
	•	•	WRITE_AS_ONE_ITEMS	Elément lexical (voir 7.4.1.2.21).
	•	•	CHARTS	Elément lexical (voir 7.4.1.2.22).
•	•	•	charts-element	Les éléments individuels de CHARTS peuvent être modifiés.
	•	•	FILES	Elément lexical (voir 7.4.1.2.28).
•	•	•	files-element	Les éléments individuels de FILES peuvent être modifiés.
	•	•	LISTS	Elément lexical (voir 7.4.1.2.23).
•	•	•	lists-element	Les éléments individuels de LISTS peuvent être modifiés.
	•	•	GRAPHS	Elément lexical (voir 7.4.1.2.24).
•	•	•	graphs-element	Les éléments individuels de GRAPH peuvent être modifiés.
	•	•	GRIDS	Elément lexical (voir 7.4.1.2.25).
•	•	•	grids-element	Les éléments individuels de GRID peuvent être modifiés.
	•	•	MENUS	Elément lexical (voir 7.4.1.2.26).
•	•	•	menus-element	Les éléments individuels de MENU peuvent être modifiés.
	•	•	METHODS	Elément lexical (voir 7.4.1.2.27).
•	•	•	methods-element	Les éléments individuels de METHODS peuvent être modifiés.
	•	•	PLUGINS	Elément lexical (voir 7.4.1.2.30).
•	•	•	plugins-element	Les éléments individuels de PLUGIN peuvent être modifiés.

7.19.2.5 BLOCK_B

Structure lexicale

BLOCK identifieur, [REDEFINE, [NUMBER, TYPE]]+

Les règles de redéfinition pour les attributs BLOCK_B sont spécifiées dans le Tableau 103 et le Tableau 107.

Tableau 107 – Règles de redéfinition pour les attributs BLOCK_B

A	D	R	Attribut	Description
		•	NUMBER	Élément lexical (voir 7.4.2.2.1).
		•	TYPE	Élément lexical (voir 7.4.2.2.2).

7.19.2.6 CHART

Structure lexicale

CHART identifieur, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, CYCLE_TIME, HEIGHT, HELP, LABEL, LENGTH, TYPE, VALIDITY, VISIBILITY, WIDTH]]+

Les règles de redéfinition pour les attributs CHART sont spécifiées dans le Tableau 103 et le Tableau 108.

Tableau 108 – Règles de redéfinition pour les attributs CHART

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.36.12).
•	•	•	members-list-element	La liste des éléments individuels de MEMBERS peut être modifiée.
	•	•	CYCLE_TIME	Élément lexical (voir 7.5.2.1).
	•	•	HEIGHT	Élément lexical (voir 7.36.7).
	•	•	HELP	Élément lexical (voir 7.36.8).
	•	•	LABEL	Élément lexical (voir 7.36.9).
	•	•	LENGTH	Élément lexical (voir 7.5.2.2).
	•	•	TYPE	Élément lexical (voir 7.5.2.3).
	•	•	VALIDITY	Élément lexical (voir 7.36.16).
	•	•	VISIBILITY	Élément lexical (voir 7.36.19).
	•	•	WIDTH	Élément lexical (voir 7.36.17).

7.19.2.7 COLLECTION

Structure lexicale

COLLECTION identifieur, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]]+

Les règles de redéfinition pour les attributs COLLECTION sont spécifiées dans le Tableau 103 et le Tableau 109.

Tableau 109 – Règles de redéfinition pour les attributs COLLECTION

A	D	R	Attribut	Description
		•	MEMBERS	Elément lexical (voir 7.36.12).
•	•	•	members-list-element	La liste des éléments individuels de MEMBERS peut être modifiée.
		•	HELP	Elément lexical (voir 7.36.8).
		•	LABEL	Elément lexical (voir 7.36.9).
		•	PRIVATE	Elément lexical (voir 7.36.18).
		•	VALIDITY	Elément lexical (voir 7.36.16).
		•	VISIBILITY	Elément lexical (voir 7.36.19).

7.19.2.8 COMMAND**Structure lexicale**

COMMAND *identif*ier, [[ADD, DELETE, REDEFINE], [OPERATION, TRANSACTION, INDEX, BLOCK_B, NUMBER, SLOT, SUB_SLOT, API, HEADER, RESPONSE_CODES, response-codes-list-element]]+

Les règles de redéfinition pour les attributs COMMAND sont spécifiées dans le Tableau 103 et le Tableau 110.

Tableau 110 – Règles de redéfinition pour les attributs COMMAND

A	D	R	Attribut	Description
			OPERATION	Elément lexical (voir 7.7.2.1).
•	•	•	TRANSACTION	Elément lexical (voir 7.7.2.2).
		•	INDEX	Elément lexical (voir 7.7.2.3).
			BLOCK_B	Elément lexical (voir 7.7.2.4).
			NUMBER	Elément lexical (voir 7.7.2.5).
		•	SLOT	Elément lexical (voir 7.7.2.6).
		•	SUB_SLOT	Elément lexical (voir 7.7.2.7).
		•	API	Elément lexical (voir 7.7.2.7).
		•	HEADER	Elément lexical (voir 7.7.2.9).
		•	POST_RECEIVE_ACTIONS	Elément lexical (voir 7.7.2.12).
		•	RESPONSE_CODES	Elément lexical (voir 7.36.14).
•	•	•	response-codes-list-element	La liste des éléments individuels de RESPONSE_CODES peut être modifiée.

7.19.2.9 COMPONENT**Structure lexicale**

COMPONENT *identif*ier, [[DELETE, REDEFINE], [LABEL, BYTE_ORDER, CAN_DELETE, CHECK_CONFIGURATION, CLASSIFICATION, CONNECTION_POINT, HELP, COMPONENT_PARENT, COMPONENT_PATH, COMPONENT_RELATIONS, DECLARATION, DETECT, EDD, INITIAL_VALUES, PRODUCT_URI, PROTOCOL, REDUNDANCY, SCAN, SCAN_LIST, SUPPLIED_INTERFACE]]+

Les règles de redéfinition pour les attributs COMPONENT sont spécifiées dans le Tableau 103 et le Tableau 111.

Tableau 111 – Règles de redéfinition pour les attributs COMPONENT

A	D	R	Attribut	Description
		•	LABEL	Elément lexical (voir 7.36.9).
	•	•	BYTE_ORDER	Elément lexical (voir 7.8.2.11).
	•	•	CAN_DELETE	Elément lexical (voir 7.8.2.1).
	•	•	CHECK_CONFIGURATION	Elément lexical (voir 7.8.2.2).
	•	•	CLASSIFICATION	Elément lexical (voir 7.36.1).
	•	•	COMPONENT_PARENT	Elément lexical (voir 7.36.2).
	•	•	COMPONENT_PATH	Elément lexical (voir 7.36.3).
	•	•	COMPONENT_RELATIONS	Elément lexical (voir 7.8.2.3).
	•	•	CONNECTION_POINT	Elément lexical (voir 7.8.2.12).
	•	•	DECLARATION	Elément lexical (voir 7.8.2.4).
	•	•	DETECT	Elément lexical (voir 7.8.2.5).
	•	•	EDD	Elément lexical (voir 7.8.2.6).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	INITIAL_VALUES	Elément lexical (voir 7.8.2.7).
	•	•	PRODUCT_URI	Elément lexical (voir 7.8.2.13).
	•	•	PROTOCOL	Elément lexical (voir 7.36.13).
	•	•	REDUNDANCY	Elément lexical (voir 7.8.2.8).
	•	•	SCAN	Elément lexical (voir 7.8.2.9).
	•	•	SCAN_LIST	Elément lexical (voir 7.8.2.10).
	•	•	SUPPLIED_INTERFACE	Elément lexical (voir 7.36.15).

7.19.2.10 COMPONENT_FOLDER

Structure lexicale

COMPONENT_FOLDER identifiant, [[DELETE, REDEFINE], [LABEL, CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, HELP, PROTOCOL]]+

Les règles de redéfinition pour les attributs COMPONENT_FOLDER sont spécifiées dans le Tableau 103 et le Tableau 112.

Tableau 112 – Règles de redéfinition pour les attributs COMPONENT_FOLDER

A	D	R	Attribut	Description
		•	LABEL	Elément lexical (voir 7.36.9).
	•	•	CLASSIFICATION	Elément lexical (voir 7.36.1).
	•	•	COMPONENT_PARENT	Elément lexical (voir 7.36.2).
	•	•	COMPONENT_PATH	Elément lexical (voir 7.36.3).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	PROTOCOL	Elément lexical (voir 7.36.13).

7.19.2.11 COMPONENT_REFERENCE

Structure lexicale

COMPONENT_REFERENCE *identifieur*, [[DELETE, REDEFINE], [CLASSIFICATION, COMPONENT_PARENT, COMPONENT_PATH, DEVICE_REVISION, DEVICE_TYPE, MANUFACTURER, PROTOCOL]]+

Les règles de redéfinition pour les attributs COMPONENT_REFERENCE sont spécifiées dans le Tableau 103 et le Tableau 113.

Tableau 113 – Règles de redéfinition pour les attributs COMPONENT_REFERENCE

A	D	R	Attribut	Description
	•	•	CLASSIFICATION	Élément lexical (voir 7.36.1).
	•	•	COMPONENT_PARENT	Élément lexical (voir 7.36.2).
	•	•	COMPONENT_PATH	Élément lexical (voir 7.36.3).
	•	•	DEVICE_REVISION	Élément lexical (voir 7.2.2.2).
	•	•	DEVICE_TYPE	Élément lexical (voir 7.2.2.3).
	•	•	MANUFACTURER	Élément lexical (voir 7.2.2.6).
	•	•	PROTOCOL	Élément lexical (voir 7.36.13).

7.19.2.12 COMPONENT_RELATION

Structure lexicale

COMPONENT_RELATION *identifieur*, [[ADD, DELETE, REDEFINE], [COMPONENTS, AUTO_CREATE, FILTER, REQUIRED_RANGES, LABEL, RELATION_TYPE, ADDRESSING, HELP, MAXIMUM_NUMBER, MINIMUM_NUMBER, REQUIRED_INTERFACE, SUPPLIED_INTERFACE]]+

Les règles de redéfinition pour les attributs COMPONENT_RELATION sont spécifiées dans le Tableau 103 et le Tableau 114.

Tableau 114 – Règles de redéfinition pour les attributs COMPONENT_RELATION

A	D	R	Attribut	Description
		•	COMPONENTS	Élément lexical (voir 7.11.2.1).
•	•	•	composants-list-element	Les éléments individuels de COMPONENTS peuvent être modifiés.
	•	•	AUTO_CREATE	Élément lexical (voir 7.11.2.1).
	•	•	FILTER	Élément lexical (voir 7.11.2.1).
	•	•	REQUIRED_RANGES	Élément lexical (voir 7.11.2.1).
		•	LABEL	Élément lexical (voir 7.36.9).
		•	RELATION_TYPE	Élément lexical (voir 7.11.2.2).
	•	•	ADDRESSING	Élément lexical (voir 7.11.2.3).
	•	•	HELP	Élément lexical (voir 7.36.8).
	•	•	MAXIMUM_NUMBER	Élément lexical (voir 7.11.2.4).
	•	•	MINIMUM_NUMBER	Élément lexical (voir 7.11.2.5).
	•	•	REQUIRED_INTERFACE	Élément lexical (voir 7.11.2.6).
	•	•	SUPPLIED_INTERFACE	Élément lexical (voir 7.36.15).

7.19.2.13 EDIT_DISPLAY

Structure lexicale

EDIT_DISPLAY *identifieur*, [[DELETE, REDEFINE], [EDIT_ITEMS, LABEL, DISPLAY_ITEMS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS]]+

Les règles de redéfinition pour les attributs EDIT_DISPLAY sont spécifiées dans le Tableau 103 et le Tableau 115.

Tableau 115 – Règles de redéfinition pour les attributs EDIT_DISPLAY

A	D	R	Attribut	Description
		•	EDIT_ITEMS	Élément lexical (voir 7.14.2.1).
		•	LABEL	Élément lexical (voir 7.36.9).
	•	•	DISPLAY_ITEMS	Élément lexical (voir 7.14.2.2).
	•	•	POST_EDIT_ACTIONS	Élément lexical (voir 7.14.2.3).
	•	•	PRE_EDIT_ACTIONS	Élément lexical (voir 7.14.2.4).

7.19.2.14 FILE

Structure lexicale

FILE *identifieur*, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, HANDLING, HELP, IDENTITY, LABEL, SHARED, ON_UPDATE_ACTIONS]]+

Les règles de redéfinition pour les attributs FILE sont spécifiées dans le Tableau 103 et le Tableau 116.

Tableau 116 – Règles de redéfinition pour les attributs FILE

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.36.12).
•	•	•	members-list-element	La liste des éléments individuels de MEMBERS peut être modifiée.
	•	•	HANDLING	Élément lexical (voir 7.36.6).
	•	•	HELP	Élément lexical (voir 7.36.8).
	•	•	IDENTITY	Élément lexical (voir 7.36.21).
	•	•	LABEL	Élément lexical (voir 7.36.9).
	•	•	SHARED	Élément lexical (voir 7.15.2.1).
	•	•	ON_UPDATE_ACTIONS	Élément lexical (voir 7.15.2.2).

7.19.2.15 GRAPH

Structure lexicale

GRAPH *identifieur*, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, CYCLE_TIME, HEIGHT, HELP, LABEL, VALIDITY, VISIBILITY, WIDTH, X_AXIS]]+

Les règles de redéfinition pour les attributs GRAPH sont spécifiées dans le Tableau 103 et le Tableau 117.

Tableau 117 – Règles de redéfinition pour les attributs GRAPH

A	D	R	Attribut	Description
		•	MEMBERS	Elément lexical (voir 7.36.12).
•	•	•	members-list-element	La liste des éléments individuels de MEMBERS peut être modifiée.
	•	•	CYCLE_TIME	Elément lexical (voir 7.16.2.1).
	•	•	HEIGHT	Elément lexical (voir 7.36.7).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	VALIDITY	Elément lexical (voir 7.38.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).
	•	•	WIDTH	Elément lexical (voir 7.36.17).
	•	•	X_AXIS	Elément lexical (voir 7.16.2.2).

7.19.2.16 GRID**Structure lexicale**

GRID identifier, [[ADD, DELETE, REDEFINE], VECTORS, HANDLING, HEIGHT, HELP, LABEL, ORIENTATION, VALIDITY, VISIBILITY, WIDTH]]+

Les règles de redéfinition pour les attributs GRID sont spécifiées dans le Tableau 103 et le Tableau 118.

Tableau 118 – Règles de redéfinition pour les attributs GRID

A	D	R	Attribut	Description
		•	VECTORS	Elément lexical (voir 7.17.2.1).
	•	•	HANDLING	Elément lexical (voir 7.36.6).
	•	•	HEIGHT	Elément lexical (voir 7.36.7).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	ORIENTATION	Elément lexical (voir 7.17.2.2).
	•	•	VALIDITY	Elément lexical (voir 7.38.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).
	•	•	WIDTH	Elément lexical (voir 7.36.17).

7.19.2.17 IMAGE**Structure lexicale**

IMAGE identifier, [[DELETE, REDEFINE], PATH, HELP, LABEL, LINK, VALIDITY, VISIBILITY]]+

Les règles de redéfinition pour les attributs IMAGE sont spécifiées dans le Tableau 103 et le Tableau 119.

Tableau 119 – Règles de redéfinition pour les attributs IMAGE

A	D	R	Attribut	Description
		•	PATH	Elément lexical (voir 7.18.2.1).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	LINK	Elément lexical (voir 7.18.2.2).
	•	•	VALIDITY	Elément lexical (voir 7.38.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).

7.19.2.18 INTERFACE

Structure lexicale

INTERFACE *identifieur*, [[DELETE, REDEFINE], [DECLARATION, LABEL, HELP]]+

Les règles de redéfinition pour les attributs INTERFACE sont spécifiées dans le Tableau 103 et le Tableau 120.

Tableau 120 – Règles de redéfinition pour les attributs INTERFACE

A	D	R	Attribut	Description
		•	DECLARATION	Elément lexical (voir 7.20.2).
		•	LABEL	Elément lexical (voir 7.36.9).
	•	•	HELP	Elément lexical (voir 7.36.8).

7.19.2.19 LIST

Structure lexicale

LIST *identifieur*, [[DELETE, REDEFINE], [TYPE, CAPACITY, COUNT, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]]+

Les règles de redéfinition pour les attributs LIST sont spécifiées dans le Tableau 103 et le Tableau 121.

Tableau 121 – Règles de redéfinition pour les attributs LIST

A	D	R	Attribut	Description
		•	TYPE	Elément lexical (voir 7.22.2.1).
	•	•	CAPACITY	Elément lexical (voir 7.22.2.2).
	•	•	COUNT	Elément lexical (voir 7.22.2.3).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	PRIVATE	Elément lexical (voir 7.36.18).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).

7.19.2.20 MENU

Structure lexicale

MENU *identifieur*, [[ADD, DELETE, REDEFINE], [ITEMS, *items-list-element*, LABEL, ACCESS, HELP, EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, STYLE, VALIDITY, VISIBILITY]]+

Les règles de redéfinition pour les attributs MENU sont spécifiées dans le Tableau 103 et le Tableau 122.

Tableau 122 – Règles de redéfinition pour les attributs MENU

A	D	R	Attribut	Description
		•	ITEMS	Elément lexical (voir 7.23.2.1).
		•	LABEL	Elément lexical (voir 7.36.9).
	•	•	ACCESS	Elément lexical (voir 7.23.2.2).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	EXIT_ACTIONS	Elément lexical (voir 7.23.2.12).
	•	•	INIT_ACTIONS	Elément lexical (voir 7.23.2.13).
	•	•	POST_EDIT_ACTIONS	Elément lexical (voir 7.23.2.3).
	•	•	POST_READ_ACTIONS	Elément lexical (voir 7.23.2.4).
	•	•	POST_WRITE_ACTIONS	Elément lexical (voir 7.23.2.5).
	•	•	PRE_EDIT_ACTIONS	Elément lexical (voir 7.23.2.6).
	•	•	PRE_READ_ACTIONS	Elément lexical (voir 7.23.2.7).
	•	•	PRE_WRITE_ACTIONS	Elément lexical (voir 7.23.2.8).
	•	•	STYLE	Elément lexical (voir 7.23.2.11).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).

7.19.2.21 METHOD

Structure lexicale

METHOD *identifieur*, [[DELETE, REDEFINE], [DEFINITION, ACCESS, CLASS, LABEL, HELP, PRIVATE, VALIDITY, VISIBILITY]]+

Les règles de redéfinition pour les attributs METHOD sont spécifiées dans le Tableau 103 et le Tableau 123.

Tableau 123 – Règles de redéfinition pour les attributs METHOD

A	D	R	Attribut	Description
		•	DEFINITION	Elément lexical (voir 7.36.4).
	•	•	ACCESS	Elément lexical (voir 7.24.2.1).
		•	CLASS	Elément lexical (voir 7.24.2.2).
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	HELP	Elément lexical (voir 7.36.8).
			TYPE	Elément lexical (voir 7.24.2.3). TYPE ne doit pas être redéfini.
	•	•	PRIVATE	Elément lexical (voir 7.36.18).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).

7.19.2.22 PLUGIN**Structure lexicale**

PLUGIN *identifieur*, [[DELETE, REDEFINE], [LABEL, HELP, UUID, VALIDITY, VISIBILITY]]+

Les règles de redéfinition pour les attributs PLUGIN sont spécifiées dans le Tableau 103 et le Tableau 124.

Tableau 124 – Règles de redéfinition pour les attributs PLUGIN

A	D	R	Attribut	Description
		•	LABEL	Elément lexical (voir 7.36.9).
	•	•	HELP	Elément lexical (voir 7.36.8).
		•	UUID	Elément lexical (voir 7.42.2).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).

7.19.2.23 RECORD**Structure lexicale**

RECORD *identifieur*, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, LABEL, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]]+

Les règles de redéfinition pour les attributs RECORD sont spécifiées dans le Tableau 103 et le Tableau 125.

Tableau 125 – Règles de redéfinition pour les attributs RECORD

A	D	R	Attribut	Description
		•	MEMBERS	Elément lexical (voir 7.36.12).
•	•	•	members-list-element	La liste des éléments individuels de MEMBERS peut être modifiée.
		•	LABEL	Elément lexical (voir 7.36.9).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	PRIVATE	Elément lexical (voir 7.36.18).
	•	•	RESPONSE_CODES	Elément lexical (voir 7.36.14).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).
	•	•	WRITE_MODE	Elément lexical (voir 7.36.20).

7.19.2.24 REFERENCE_ARRAY**Structure lexicale**

REFERENCE_ARRAY *identifieur*, [[ADD, DELETE, REDEFINE], [ELEMENTS, elements-list-element, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]]+

Les règles de redéfinition pour les attributs REFERENCE_ARRAY sont spécifiées dans le Tableau 103 et le Tableau 126.

Tableau 126 – Règles de redéfinition pour les attributs REFERENCE_ARRAY

A	D	R	Attribut	Description
		•	ELEMENTS	Elément lexical (voir 7.27.2).
•	•	•	elements-list-element	La liste des éléments individuels d'ELEMENTS peut être modifiée.
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	PRIVATE	Elément lexical (voir 7.36.18).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).

7.19.2.25 RESPONSE_CODES**Structure lexicale**

RESPONSE_CODES *identifieur*, [[ADD, DELETE, REDEFINE], response-code-list-element] +

Les règles de redéfinition pour les attributs RESPONSE_CODES sont spécifiées dans le Tableau 103 et le Tableau 127.

Tableau 127 – Règles de redéfinition pour les attributs RESPONSE_CODES

A	D	R	Attribut	Description
•	•	•	response-code-list-element	La liste des éléments individuels de RESPONSE_CODES peut être modifiée.

7.19.2.26 SOURCE

Structure lexicale

SOURCE identifieur, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, EMPHASIS, EXIT_ACTIONS, HELP, INIT_ACTIONS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY, VISIBILITY, Y_AXIS]]+

Les règles de redéfinition pour les attributs SOURCE sont spécifiées dans le Tableau 103 et le Tableau 128.

Tableau 128 – Règles de redéfinition pour les attributs SOURCE

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.36.12).
•	•	•	members-list-element	La liste des éléments individuels de MEMBERS peut être modifiée.
	•	•	EMPHASIS	Élément lexical (voir 7.36.5).
	•	•	EXIT_ACTIONS	Élément lexical (voir 7.30.2.1).
	•	•	HELP	Élément lexical (voir 7.36.8).
	•	•	INIT_ACTIONS	Élément lexical (voir 7.30.2.2).
	•	•	LABEL	Élément lexical (voir 7.36.9).
	•	•	LINE_COLOR	Élément lexical (voir 7.36.10).
	•	•	LINE_TYPE	Élément lexical (voir 7.36.11).
	•	•	REFRESH_ACTIONS	Élément lexical (voir 7.30.2.3).
	•	•	VALIDITY	Élément lexical (voir 7.36.16).
	•	•	VISIBILITY	Élément lexical (voir 7.36.19).
	•	•	Y_AXIS	Élément lexical (voir 7.30.2.4).

7.19.2.27 TEMPLATE

Structure lexicale

TEMPLATE identifieur, [[DELETE, REDEFINE], [DEFAULT_VALUES, LABEL, HELP, VALIDITY]]+

Les règles de redéfinition pour les attributs TEMPLATE sont spécifiées dans le Tableau 103 et le Tableau 129.

Tableau 129 – Règles de redéfinition pour les attributs TEMPLATE

A	D	R	Attribut	Description
		•	DEFAULT_VALUES	Élément lexical (voir 7.31.2).
		•	LABEL	Élément lexical (voir 7.36.9).
	•	•	HELP	Élément lexical (voir 7.36.8).
	•	•	VALIDITY	Élément lexical (voir 7.36.16).

7.19.2.28 VALUE_ARRAY

Structure lexicale

VALUE_ARRAY identifier, [[DELETE, REDEFINE], [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]]+

Les règles de redéfinition pour les attributs VALUE_ARRAY sont spécifiées dans le Tableau 103 et le Tableau 130.

Tableau 130 – Règles de redéfinition pour les attributs VALUE_ARRAY

A	D	R	Attribut	Description
		•	LABEL	Elément lexical (voir 7.36.9).
		•	NUMBER_OF_ELEMENTS	Elément lexical (voir 7.32.2.1).
		•	TYPE	Elément lexical (voir 7.32.2.2).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	PRIVATE	Elément lexical (voir 7.36.18).
	•	•	RESPONSE_CODES	Elément lexical (voir 7.36.14).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).
	•	•	WRITE_MODE	Elément lexical (voir 7.36.20).

7.19.2.29 VARIABLE

Structure lexicale

VARIABLE identifier, [[ADD, DELETE, REDEFINE], [CLASS, LABEL, TYPE, DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, MAX_VALUE, MIN_VALUE, SCALING_FACTOR, TIME_FORMAT, TIME_SCALE, type-list-element, CONSTANT_UNIT, HANDLING, HEIGHT, HELP, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_RESET_ACTIONS, POST_UPDATE_ACTIONS, POST_USERCHANGE_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, PRIVATE, REFRESH_ACTIONS, RESPONSE_CODES, VALIDITY, VISIBILITY, WIDTH, WRITE_MODE]]+

Les règles de redéfinition pour les attributs VARIABLE sont spécifiées dans le Tableau 103 et le Tableau 131.

Tableau 131 – Règles de redéfinition pour les attributs VARIABLE

A	D	R	Attribut	Description
		•	CLASS	Elément lexical (voir 7.33.2.1).
	•	•	LABEL	Elément lexical (voir 7.36.9).
			TYPE	Elément lexical (voir 7.33.2.2). TYPE ne doit pas être redéfini.
•	•	•	type-list-element	Les éléments de liste enumeration et bit-enumeration individuels peuvent être modifiés.
	•	•	DEFAULT_VALUE	Elément lexical (voir 7.33.2.2).
	•	•	DISPLAY_FORMAT	Elément lexical (voir 7.33.2.2).
	•	•	EDIT_FORMAT	Elément lexical (voir 7.33.2.2).
	•	•	INITIAL_VALUE	Elément lexical (voir 7.33.2.2).
	•	•	MAX_VALUE	Elément lexical (voir 7.33.2.2).
	•	•	MIN_VALUE	Elément lexical (voir 7.33.2.2).
	•	•	SCALING_FACTOR	Elément lexical (voir 7.33.2.2).
	•	•	TIME_FORMAT	Elément lexical (voir 7.33.2.2).
	•	•	TIME_SCALE	Elément lexical (voir 7.33.2.2).
	•	•	CONSTANT_UNIT	Elément lexical (voir 7.33.2.3).
	•	•	HANDLING	Elément lexical (voir 7.36.6).
	•	•	HEIGHT	Elément lexical (voir 7.36.7).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	POST_EDIT_ACTIONS	Elément lexical (voir 7.33.2.6).
	•	•	POST_READ_ACTIONS	Elément lexical (voir 7.33.2.7).
	•	•	POST_RQSTUPDATE_ACTIONS	Elément lexical (voir 7.33.2.15).
	•	•	POST_USERCHANGE_ACTIONS	Elément lexical (voir 7.33.2.16).
	•	•	POST_WRITE_ACTIONS	Elément lexical (voir 7.33.2.8).
	•	•	PRE_EDIT_ACTIONS	Elément lexical (voir 7.33.2.9).
	•	•	PRE_READ_ACTIONS	Elément lexical (voir 7.33.2.10).
	•	•	PRE_WRITE_ACTIONS	Elément lexical (voir 7.33.2.11).
	•	•	PRIVATE	Elément lexical (voir 7.36.18).
	•	•	REFRESH_ACTIONS	Elément lexical (voir 7.33.2.13).
	•	•	RESPONSE_CODES	Elément lexical (voir 7.36.14).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).
	•	•	WIDTH	Elément lexical (voir 7.36.17).
	•	•	WRITE_MODE	Elément lexical (voir 7.36.20).

7.19.2.30 VARIABLE_LIST**Structure lexicale**

VARIABLE_LIST `identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, HELP, LABEL, RESPONSE_CODES]]+`

Les règles de redéfinition pour les attributs VARIABLE_LIST sont spécifiées dans le Tableau 103 et le Tableau 132.

Tableau 132 – Règles de redéfinition pour les attributs VARIABLE_LIST

A	D	R	Attribut	Description
		•	MEMBERS	Elément lexical (voir 7.36.12).
•	•	•	members-list-element	La liste des éléments individuels de MEMBERS peut être modifiée.
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	RESPONSE_CODES	Elément lexical (voir 7.36.14).

7.19.2.31 WAVEFORM**Structure lexicale**

WAVEFORM `identifieur, [[ADD, DELETE, REDEFINE], [TYPE, X_VALUES, Y_VALUES, NUMBER_OF_POINTS, X_INITIAL, X_INCREMENT, EMPHASIS, HANDLING, HELP, INIT_ACTIONS, KEY_POINTS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY, VISIBILITY, Y_AXIS]]+`

Les règles de redéfinition pour les attributs WAVEFORM sont spécifiées dans le Tableau 103 et le Tableau 133.

Tableau 133 – Règles de redéfinition pour les attributs WAVEFORM

A	D	R	Attribut	Description
		•	TYPE	Elément lexical (voir 7.35.2.1).
•		•	X_VALUES	Elément lexical (voir 7.35.2.1).
•		•	Y_VALUES	Elément lexical (voir 7.35.2.1).
	•	•	NUMBER_OF_POINTS	Elément lexical (voir 7.35.2.1).
		•	X_INITIAL	Elément lexical (voir 7.35.2.1).
		•	X_INCREMENT	Elément lexical (voir 7.35.2.1).
	•	•	EMPHASIS	Elément lexical (voir 7.36.5).
	•	•	EXIT_ACTIONS	Elément lexical (voir 7.35.2.2).
	•	•	HANDLING	Elément lexical (voir 7.36.6).
	•	•	HELP	Elément lexical (voir 7.36.8).
	•	•	INIT_ACTIONS	Elément lexical (voir 7.35.2.3).
	•	•	KEY_POINTS	Elément lexical (voir 7.35.2.4).
	•	•	LABEL	Elément lexical (voir 7.36.9).
	•	•	LINE_COLOR	Elément lexical (voir 7.36.10).
	•	•	LINE_TYPE	Elément lexical (voir 7.36.11).
	•	•	REFRESH_ACTIONS	Elément lexical (voir 7.35.2.5).
	•	•	VALIDITY	Elément lexical (voir 7.36.16).
	•	•	VISIBILITY	Elément lexical (voir 7.36.19).
	•	•	Y_AXIS	Elément lexical (voir 7.35.2.6).

7.20 INTERFACE

7.20.1 Structure générale

Objet

L'attribut INTERFACE spécifie un ensemble de constructions de base de composants qui sont accessibles par d'autres composants. Une interface peut contenir des déclarations de constructions de base, par exemple, des attributs VARIABLE, MENU.

Il convient que tous les attributs INTERFACE fournis d'un composant soient répertoriés dans l'attribut SUPPLIED_INTERFACE du COMPONENT.

Si des attributs INTERFACE sont définis pour un attribut COMPONENT, l'accès est restreint. D'autres composants peuvent accéder uniquement aux constructions de base de l'EDD (par exemple, les attributs VARIABLE) définies dans une interface.

Si aucune interface n'est définie pour le composant, toutes les constructions de base peuvent y accéder.

Structure lexicale

INTERFACE identifier [LABEL, HELP, DECLARATION]

Les attributs INTERFACE sont spécifiés dans le Tableau 134.

Tableau 134 – Attributs INTERFACE

Utilisa-tion	Attribut	Description
m	DECLARATION	Élément lexical (voir 7.20.2).
m	LABEL	Élément lexical (voir 7.36.9).
o	HELP	Élément lexical (voir 7.36.8).

7.20.2 Attribut spécifique – DECLARATION

Objet

L'attribut DECLARATION spécifie les constructions de base incluses dans l'interface.

Structure lexicale

DECLARATION [IMAGE, MENU, METHOD, VARIABLE] +

Les attributs DECLARATION sont spécifiés dans le Tableau 135.

Tableau 135 – Attributs DECLARATION

Utilisa-tion	Attribut	Description
o	IMAGE	Déclarations d'IMAGE (voir 7.18).
o	MENU	Déclarations de MENU (voir 7.23).
o	METHOD	Déclarations de METHOD (voir 7.24).
o	VARIABLE	Déclarations de VARIABLE (voir 7.33).

7.21 LIKE

Objet

LIKE est utilisé pour créer une nouvelle instance d'EDD sur la base d'un élément existant. LIKE fait une copie du type d'élément existant avec tous les attributs.

Structure lexicale

LIKE *identifieur*, *reference*, *attribute-redefinition*+, *item-type*

Les attributs LIKE sont spécifiés dans le Tableau 136.

Tableau 136 – Attributs LIKE

Utilisa-tion	Attribut	Description
m	identifieur	Nom de la nouvelle instance d'EDD.
m	reference	Référence à une instance d'EDD qui doit être copiée.
o	attribute-redefinition	Spécifie si un attribut est ajouté, supprimé ou redéfini (voir 7.19.2). Seuls les attributs existants peuvent être supprimés ou redéfinis.
o	item-type	Type de la nouvelle instance d'EDD. S'il est spécifié, il doit correspondre au type de l'élément référencé.

7.22 LIST

7.22.1 Structure générale

Objet

LIST décrit une séquence d'éléments dans laquelle chaque élément de la séquence est défini par une instance de construction EDDL.

Structure lexicale

LIST *identifieur* [TYPE, CAPACITY, COUNT, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]

Les attributs LIST sont spécifiés dans le Tableau 137.

Tableau 137 – Attributs LIST

Utilisa-tion	Attribut	Description
m	TYPE	Élément lexical (voir 7.22.2.1).
o	CAPACITY	Élément lexical (voir 7.22.2.2).
o	COUNT	Élément lexical (voir 7.22.2.3).
o	HELP	Élément lexical (voir 7.36.8).
o	LABEL	Élément lexical (voir 7.36.9).
o	PRIVATE	Élément lexical (voir 7.36.18).
o	VALIDITY	Élément lexical (voir 7.36.16).
o	VISIBILITY	Élément lexical (voir 7.36.19).

7.22.2 Attributs spécifiques

7.22.2.1 TYPE

Objet

L'attribut TYPE est une référence à une instance de construction EDDL (par exemple, VARIABLE, COLLECTION). Tous les attributs de la construction EDDL s'appliquent à chacun des éléments de l'attribut LIST.

Structure lexicale

TYPE reference

L'attribut TYPE est spécifié dans le Tableau 138.

Tableau 138 – Attribut TYPE

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de VARIABLE, une instance de RECORD, une instance de VALUE_ARRAY, une instance de COLLECTION, une instance de REFERENCE_ARRAY ou une instance de LIST. L'élément référence dans l'attribut TYPE d'un attribut LIST ne doit pas contenir d'attribut VALIDITY ou toute autre expression conditionnelle qui provoquerait des modifications de la structure du type des données. Cette restriction s'applique aussi aux éléments contenus. La référence doit se terminer sur un élément de données.

7.22.2.2 CAPACITY

Objet

L'attribut CAPACITY spécifie le nombre maximal d'éléments qui peuvent être stockés dans la liste.

Structure lexicale

CAPACITY unsigned_integer

L'attribut CAPACITY est spécifié dans le Tableau 139.

Tableau 139 – Attribut CAPACITY

Utilisa-tion	Attribut	Description
m	unsigned_integer	Constante entière non signée supérieure ou égale à un.

7.22.2.3 COUNT

Objet

L'attribut COUNT spécifie le nombre d'éléments actuellement stockés dans la liste. Lorsque LIST fait partie d'un attribut FILE, COUNT ne doit pas être spécifié étant donné que seule l'application EDD connaît le nombre d'éléments actuellement présents dans la liste.

Si COUNT est spécifié, il définit la taille lors de la création de l'instance de LIST. S'il n'est pas défini, une instance de LIST avec zéro élément est créée.

Structure lexicale

COUNT (unsigned_integer) <exp>

L'attribut COUNT est spécifié dans le Tableau 140.

Tableau 140 – Attribut COUNT

Utilisa-tion	Attribut	Description
m	unsigned_integer	Constante entière non signée.

7.23 MENU

7.23.1 Structure générale

Objet

MENU organise dans une structure hiérarchique des variables, des méthodes et d'autres éléments spécifiés dans le langage EDDL. Une application utilisateur peut utiliser les attributs ITEMS de MENU pour afficher et entrer des informations d'une façon organisée et cohérente.

Structure lexicale

MENU identifieur [ITEMS, LABEL, ACCESS, HELP, EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, STYLE, VALIDITY, VISIBILITY]

Les attributs MENU sont spécifiés dans le Tableau 141.

Tableau 141 – Attributs MENU

Utilisa-tion	Attribut	Description
m	ITEMS	Elément lexical (voir 7.23.2.1).
m	LABEL	Elément lexical (voir 7.36.9).
o	ACCESS	Elément lexical (voir 7.23.2.2).
o	HELP	Elément lexical (voir 7.36.8).
o	EXIT_ACTIONS	Elément lexical (voir 7.23.2.12).
o	INIT_ACTIONS	Elément lexical (voir 7.23.2.13).
o	POST_EDIT_ACTIONS	Elément lexical (voir 7.23.2.3).
o	POST_READ_ACTIONS	Elément lexical (voir 7.23.2.4).
o	POST_WRITE_ACTIONS	Elément lexical (voir 7.23.2.5).
o	PRE_EDIT_ACTIONS	Elément lexical (voir 7.23.2.6).
o	PRE_READ_ACTIONS	Elément lexical (voir 7.23.2.7).
o	PRE_WRITE_ACTIONS	Elément lexical (voir 7.23.2.8).
o	STYLE	Elément lexical (voir 7.23.2.11).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).

7.23.2 Attributs spécifiques

7.23.2.1 ITEMS

Objet

L'attribut ITEMS spécifie les éléments sélectionnés de l'EDD qui doivent être affichés à l'utilisateur, ainsi que des qualificatifs facultatifs pour chaque élément, pour contrôler le traitement de l'élément.

NOTE 1 Les éléments de MENU sont présentés à l'utilisateur dans l'ordre dans lequel ils apparaissent.

NOTE 2 Pour des menus verticaux, le premier élément est affiché en haut et le dernier élément en bas. Pour des menus horizontaux, le premier élément est affiché sur la gauche et le dernier élément sur la droite.

COLUMNBREAK et ROWBREAK ne doivent pas apparaître dans une instruction conditionnelle.

Structure lexicale

```
ITEMS [(reference, (DISPLAY_VALUE, HIDDEN, READ_ONLY, NO_LABEL,
NO_UNIT, REVIEW, INLINE, ALIGN_LEFT, ALIGN_RIGHT)<exp>, SEPARATOR,
ROWBREAK, COLUMNBREAK, string)]+
```

Les attributs ITEMS sont spécifiés dans le Tableau 142.

Tableau 142 – Attributs ITEMS

Utilisa-tion	Attribut	Description
m	reference	Référence à un attribut VARIABLE, un bit d'un BIT_ENUMERATED VARIABLE, BLOCK_A PARAMETERS, un élément de BLOCK_A PARAMETERS, CHART, COLLECTION, EDIT_DISPLAY, GRAPH, GRID, IMAGE, LIST, MENU, METHOD, attribut METHOD avec des arguments, PLUGIN, RECORD, REFERENCE_ARRAY ou VALUE_ARRAY.
c/o	DISPLAY_VALUE	Ne doit être utilisé qu'avec VARIABLE. Spécifie que la valeur de VARIABLE correspondante est affichée en plus de son attribut LABEL.
o	HIDDEN	L'élément n'est pas affiché.
c/o	READ_ONLY	Ne doit être utilisé qu'avec VARIABLE. Spécifie que la valeur doit être affichée mais qu'elle peut ne pas être modifiée (indépendamment de son attribut HANDLING).
c/o	NO_LABEL	Ne doit être utilisé qu'avec VARIABLE. Spécifie que l'étiquette de VARIABLE n'est pas affichée; DISPLAY_VALUE est implicite.
c/o	NO_UNIT	Ne doit être utilisé qu'avec VARIABLE. Spécifie que l'unité technique de VARIABLE n'est pas affichée; DISPLAY_VALUE est implicite.
c/o	REVIEW	Ne doit être utilisé qu'avec MENU. REVIEW est utilisé pour présenter une grande quantité de données, par exemple, pour les passer en revue.
c/o	INLINE	Ne doit être utilisé qu'avec IMAGE. Si IMAGE est dans l'attribut MENU avec STYLE égal à WINDOW, DIALOG, PAGE ou GROUP et si le qualificatif INLINE n'est pas spécifié, il convient qu'IMAGE couvre l'étendue de MENU; sinon, il convient qu'IMAGE soit affichée en ligne avec les autres attributs ITEMS de MENU.
c/o	ALIGN_LEFT	Ne doit être utilisé qu'avec IMAGE. Si IMAGE est dans l'attribut MENU avec STYLE égal à WINDOW, DIALOG, PAGE ou GROUP et si le qualificatif ALIGN_LEFT est spécifié, il convient qu'IMAGE s'affiche alignée à gauche. Si aucun alignement n'est spécifié, il convient de centrer IMAGE.

Utilisa-tion	Attribut	Description
c/o	ALIGN_RIGHT	Ne doit être utilisé qu'avec IMAGE. Si IMAGE est dans l'attribut MENU avec STYLE égal à WINDOW, DIALOG, PAGE ou GROUP et si le qualificatif ALIGN_RIGHT est spécifié, il convient qu'IMAGE s'affiche alignée à droite. Si aucun alignement n'est spécifié, il convient de centrer IMAGE.
c/o	SEPARATOR	Il convient de ne l'utiliser que dans l'attribut MENU dont l'attribut STYLE est MENU. Il convient que les éléments de menu avant et après SEPARATOR soient séparés, par exemple, par une ligne horizontale.
c/o	COLUMNBREAK	Ne doit être utilisé que dans l'attribut MENU dont STYLE correspond à WINDOW, DIALOG, PAGE ou GROUP; spécifie qu'il convient qu'une nouvelle colonne commence.
c/o	ROWBREAK	Ne doit être utilisé que dans l'attribut MENU dont STYLE correspond à WINDOW, DIALOG, PAGE ou GROUP; spécifie qu'il convient qu'une nouvelle ligne commence.
o	string	Texte statique qu'il convient d'afficher sur le menu.

7.23.2.2 ACCESS

Objet

L'attribut ACCESS spécifie si les éléments affichés ou utilisés dans l'attribut MENU sont lus depuis l'appareil (ONLINE) ou localement (OFFLINE). Si aucun attribut ACCESS n'est défini, l'attribut ACCESS de l'attribut MENU parent doit être appliqué.

Structure lexicale

ACCESS (OFFLINE, ONLINE)

Les attributs ACCESS sont spécifiés dans le Tableau 143.

Tableau 143 – Attributs ACCESS

Utilisa-tion	Attribut	Description
s	OFFLINE	Les valeurs des éléments affichés dans l'attribut MENU sont lues localement.
s	ONLINE	Il convient que les valeurs des éléments affichés dans MENU soient lues depuis l'appareil et réactualisées à un intervalle approprié.

NOTE L'implémentation est responsable de décider à quel moment effectuer le transfert de valeurs modifiées vers un appareil (typiquement après l'exécution d'une entrée unique ou un attribut MENU pour un attribut ONLINE ACCESS ou via un menu séparé pour mettre à jour l'appareil).

7.23.2.3 POST_EDIT_ACTIONS

Objet

L'attribut POST_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après que l'utilisateur a fini le traitement de MENU. Si l'attribut MENU est abandonné, les attributs POST_EDIT_ACTIONS ne sont pas exécutés. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Les attributs POST_EDIT_ACTIONS éventuellement associés à une entité référencée dans les attributs ITEMS doivent être exécutés lorsqu'une nouvelle valeur est entrée pour l'entité. Les attributs POST_EDIT_ACTIONS de MENU sont exécutés uniquement après que tous les POST_EDIT_ACTIONS spécifiques à l'entité sont terminés.

L'attribut POST_EDIT_ACTIONS ne doit pas être exprimé en utilisant des instructions conditionnelles IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

POST_EDIT_ACTIONS [reference]+, DEFINITION

Les attributs POST_EDIT_ACTIONS sont spécifiés dans le Tableau 144.

Tableau 144 – Attributs EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS

Utilisation	Attribut	Description
o	reference	Référence à une instance de METHOD ou à une instance de METHOD avec arguments.
o	DEFINITION	Élément lexical (voir 7.36.4).

7.23.2.4 POST_READ_ACTIONS

Objet

L'attribut POST_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après que toutes les entités référencées dans l'attribut ITEMS ont été lues et traitées. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés et l'activation de MENU est annulée.

Les attributs POST_READ_ACTIONS éventuellement associés à une entité référencée dans les attributs ITEMS doivent être exécutés lorsqu'une nouvelle valeur est lue pour l'entité. Les attributs POST_READ_ACTIONS de MENU doivent être exécutés uniquement après que tous les attributs POST_READ_ACTIONS spécifiques à l'entité sont terminés.

L'attribut POST_READ_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

POST_READ_ACTIONS [reference]+, DEFINITION

Les attributs POST_READ_ACTIONS sont spécifiés dans le Tableau 144.

7.23.2.5 POST_WRITE_ACTIONS

Objet

L'attribut POST_WRITE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après que toutes les entités référencées dans l'attribut ITEMS ont été écrites et traitées. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Les attributs POST_WRITE_ACTIONS éventuellement associés à une entité référencée dans les attributs ITEMS doivent être exécutés lorsqu'une nouvelle valeur est écrite pour l'entité. Les attributs POST_WRITE_ACTIONS de MENU doivent être exécutés après que tous les POST_WRITE_ACTIONS spécifiques à l'entité sont terminés.

L'attribut POST_WRITE_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

POST_WRITE_ACTIONS [reference]+, DEFINITION

Les attributs POST_WRITE_ACTIONS sont spécifiés dans le Tableau 144.

7.23.2.6 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement après l'activation de MENU. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés et l'activation de MENU est annulée.

Les attributs PRE_EDIT_ACTIONS de MENU doivent être exécutés avant l'un quelconque des attributs PRE_EDIT_ACTIONS définis des entités référencées. Une fois que les attributs PRE_EDIT_ACTIONS de MENU sont terminés, les attributs PRE_EDIT_ACTIONS éventuels pour les entités référencées doivent être exécutés.

L'attribut PRE_EDIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

PRE_EDIT_ACTIONS [reference]+, DEFINITION

Les attributs PRE_EDIT_ACTIONS sont spécifiés dans le Tableau 144.

7.23.2.7 PRE_READ_ACTIONS

Objet

L'attribut PRE_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant que toutes les entités référencées dans l'attribut ITEMS ne soient lues ou traitées. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés et l'activation de MENU est annulée. Une fois que les attributs PRE_READ_ACTIONS de MENU sont terminés, les attributs PRE_READ_ACTIONS éventuels pour les entités référencées doivent être exécutés.

L'attribut PRE_READ_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

PRE_READ_ACTIONS [reference]+, DEFINITION

Les attributs PRE_READ_ACTIONS sont spécifiés dans le Tableau 144.

7.23.2.8 PRE_WRITE_ACTIONS

Objet

L'attribut PRE_WRITE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant que toutes les entités référencées dans l'attribut ITEMS n'aient été écrites et traitées. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés. Une fois que les attributs PRE_WRITE_ACTIONS de MENU sont terminés, les attributs PRE_WRITE_ACTIONS pour les entités référencées doivent être exécutés.

L'attribut PRE_WRITE_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle (IF, IF-ELSE, SELECT). Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

PRE_WRITE_ACTIONS [reference]+, DEFINITION

Les attributs PRE_WRITE_ACTIONS sont spécifiés dans le Tableau 144.

7.23.2.9 PURPOSE (vide)

7.23.2.10 ROLE (vide)

7.23.2.11 STYLE

Objet

L'attribut STYLE spécifie la manière dont l'attribut MENU est affiché.

Lorsque l'attribut STYLE est DIALOG, WINDOW, PAGE ou GROUP, le qualificatif DISPLAY_VALUE (voir 7.23.2.1) est appliqué à chaque élément sur l'attribut MENU.

Si un ou plusieurs MENU sont actifs et un MENU DIALOG est activé, l'attribut MENU DIALOG est prioritaire et doit être fermé avant que tout autre MENU ne puisse être utilisé.

Structure lexicale

STYLE (DIALOG, WINDOW, PAGE, GROUP, MENU, TABLE)

Les attributs STYLE sont spécifiés dans le Tableau 145.

Tableau 145 – Attributs STYLE

Utilisa-tion	Attribut	Description
s	DIALOG	Aucun autre MENU ne peut être activé lorsque cet attribut MENU, ou l'un quelconque de ses sous-menus, est actif. L'attribut MENU doit être fermé avant que des interactions avec d'autres attributs MENU ne puissent être réalisées, c'est-à-dire qu'il convient que les éléments de MENU soient affichés sous forme de boîte de dialogue modale.
s	WINDOW	D'autres attributs MENU peuvent être activés pendant que cet attribut MENU est actif. Les interactions avec tous les attributs MENU activés ayant des attributs STYLE WINDOW peuvent être réalisées, c'est-à-dire qu'il convient que les éléments de MENU soient affichés en tant que fenêtre non modale.
s	PAGE	Il convient que les éléments de MENU soient affichés sous forme de page de propriétés à onglets.
s	GROUP	Il convient que les éléments de MENU soient affichés sous forme de cadre de groupe.
s	MENU	Il convient que les éléments de MENU soient affichés sous forme de menu déroulant ou contextuel.
s	TABLE	Il convient que les éléments de MENU soient affichés en colonnes verticales.

7.23.2.12 EXIT_ACTIONS**Objet**

L'attribut EXIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après que l'utilisateur a fini le traitement de MENU, même si MENU est abandonné. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Les attributs EXIT_ACTIONS de MENU doivent être exécutés après tous les autres attributs ACTIONS de MENU.

L'attribut EXIT_ACTIONS ne doit pas être exprimé à l'aide d'une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

```
EXIT_ACTIONS [reference]+, DEFINITION
```

Les attributs EXIT_ACTIONS sont spécifiés dans le Tableau 144.

7.23.2.13 INIT_ACTIONS**Objet**

L'attribut INIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement après l'activation de MENU. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés et l'activation de MENU est annulée.

Les attributs INIT_ACTIONS de MENU doivent être exécutés avant tous les autres attributs ACTIONS de MENU.

L'attribut INIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

INIT_ACTIONS [reference]+, DEFINITION

Les attributs INIT_ACTIONS sont spécifiés dans le Tableau 144.

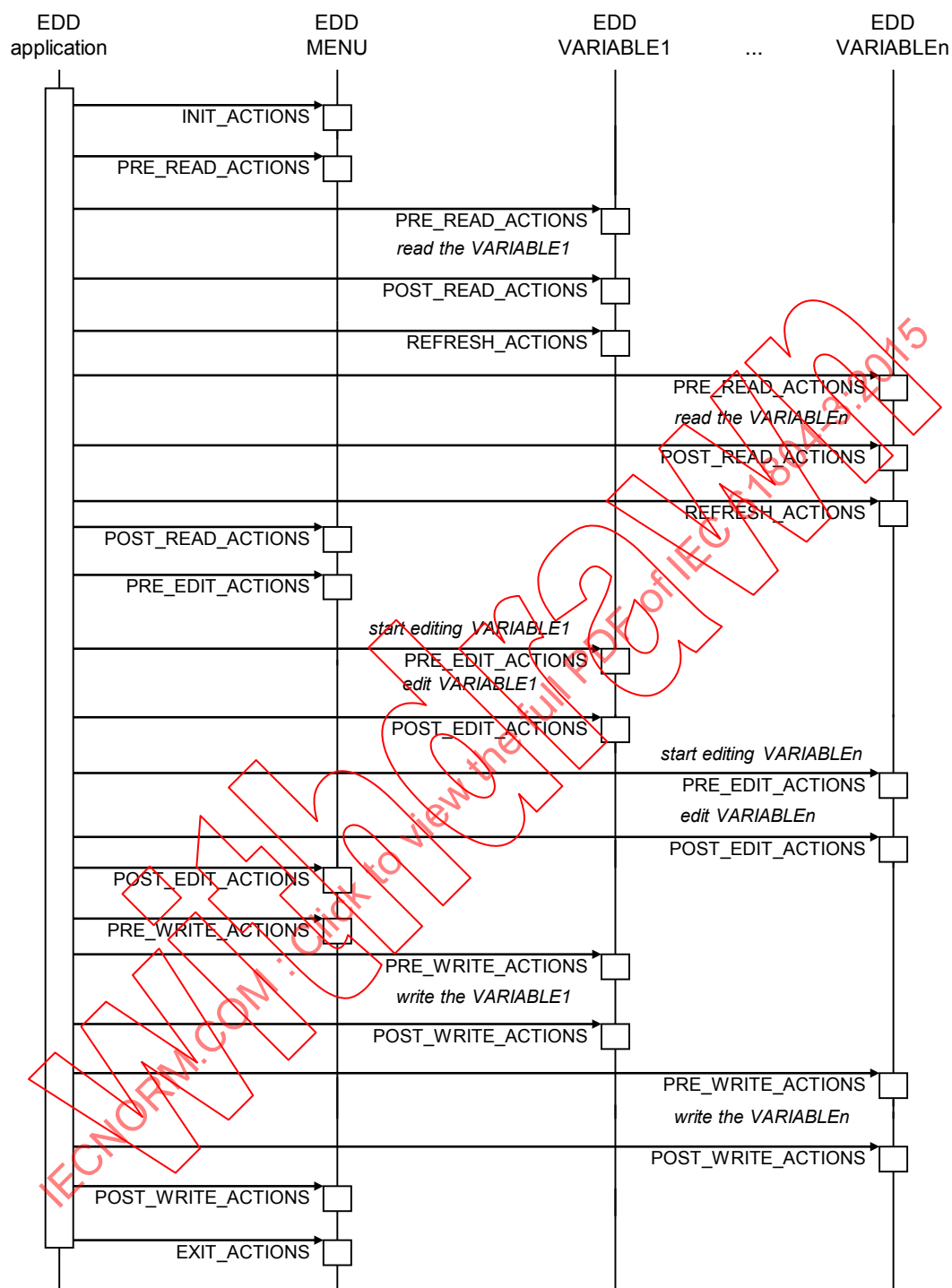
7.23.3 Schémas de séquence pour les actions

La Figure 30 présente des schémas de séquence pour des actions relatives aux attributs MENU et aux attributs VARIABLE et METHOD qu'ils contiennent. Les actions sont exécutées et terminées en séquence du haut vers le bas du schéma. Si une action échoue ou est abandonnée pour une raison imprévue, les actions restantes ne sont pas exécutées et l'attribut MENU est annulé.

Lorsque MENU a un ACCESS OFFLINE (données provenant d'une base de données locale), les actions pré/post-lecture/écriture pour l'attribut VARIABLE ne sont pas exécutées.

Lorsque l'attribut MENU a un attribut ACCESS ONLINE (données lues depuis ou écrites vers l'appareil), les actions pré/post-lecture/écriture pour VARIABLE doivent être exécutées dans l'ordre indiqué dans la séquence.

Les actions init/exit et les actions pré/post-édition doivent toujours être exécutées en cas d'ACCESS OFFLINE et ONLINE.



IEC

Anglais	Français
EDD application	Application EDD
EDD MENU	MENU EDD
EDD VARIABLE1	VARIABLE1 EDD
EDD VARIABLEn	VARIABLEn EDD
Read the VARIABLE1	Lire VARIABLE1
Read the VARIABLEn	Lire VARIABLEn
Start editing VARIABLE1	Commencer à modifier VARIABLE1

Edit VARIABLE1	Modifier VARIABLE1
Start editing VARIABLEn	Commencer à modifier VARIABLEn
Edit VARIABLEn	Modifier VARIABLEn
Write the VARIABLE1	Ecrire VARIABLE1
Write the VARIABLEn	Ecrire VARIABLEn

Figure 30 – Activation de MENU

7.24 METHOD

7.24.1 Structure générale

Objet

Un attribut METHOD est utilisé pour définir un sous-programme qui est exécuté dans l'application EDD.

NOTE Un attribut METHOD est utilisé pour spécifier des contrôles de cohérence, des algorithmes de conversion entre des variables de l'EDD, des accusés de réception d'utilisateur ou des interactions d'appareil spécifiques.

Structure lexicale

METHOD `identifieur` [(parameter-type parameter-name)+] [DEFINITION, ACCESS, CLASS, HELP, LABEL, PRIVATE, TYPE, VALIDITY, VISIBILITY]

Les attributs METHOD sont spécifiés dans le Tableau 146.

Tableau 146 – Attributs METHOD

Utilisa-tion	Attribut	Description
m	DEFINITION	Elément lexical (voir 7.36.4).
o	ACCESS	Elément lexical (voir 7.24.2.1).
o	CLASS	Elément lexical (voir 7.24.2.2).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	PRIVATE	Elément lexical (voir 7.36.18).
o	TYPE	Elément lexical (voir 7.24.2.3).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	parameter-type	Type du paramètre (voir Tableau 147).
o	parameter-name	Nom du paramètre sous la forme d'un identificateur.

Les types envisageables de paramètres sont spécifiés dans le Tableau 147.

Tableau 147 – Types de paramètres

Utilisa-tion	Attribut	Description
s	char	METHOD attend un paramètre de type caractère.
s	unsigned char	METHOD attend un paramètre de type caractère non signé.
s	short	METHOD attend un paramètre de type court.
s	unsigned short	METHOD attend un paramètre de type court non signé.
s	int	METHOD attend un paramètre de type entier.
s	unsigned int	METHOD attend un paramètre de type entier non signé.
s	long	METHOD attend un paramètre de type long.
s	unsigned long	METHOD attend un paramètre de type long non signé.
s	long long	METHOD attend un paramètre de type long long.
s	unsigned long long	METHOD attend un paramètre de type long long non signé.
s	float	METHOD attend un paramètre de type à virgule flottante.
s	double	METHOD attend un paramètre de type à double précision.
s	time_t	METHOD attend un paramètre de type time_t.
s	DD_STRING	METHOD attend un paramètre de type DD_STRING.
s	DD_ITEM	METHOD attend un paramètre de type DD_ITEM.

7.24.2 Attributs spécifiques

7.24.2.1 ACCESS

Objet

L'attribut ACCESS spécifie si les valeurs de VARIABLE stockées dans l'appareil ou l'ensemble de données hors ligne sont utilisées dans l'attribut DEFINITION de METHOD. Si ACCESS n'est pas défini, la valeur par défaut est ONLINE.

Structure lexicale

ACCESS (OFFLINE, ONLINE)

Les attributs ACCESS sont spécifiés dans le Tableau 148.

Tableau 148 – Attributs ACCESS

Utilisa-tion	Attribut	Description
s	OFFLINE	Les valeurs des variables utilisées dans la définition de METHOD sont lues ou écrites depuis/vers l'ensemble de données hors connexion.
s	ONLINE	Les valeurs des variables utilisées dans la définition de METHOD sont effectivement lues depuis l'appareil. Les changements de valeurs de VARIABLE sont écrits avec des Builtins.

7.24.2.2 CLASS

Objet

L'attribut CLASS spécifie l'effet d'un attribut METHOD sur un appareil. Cet attribut est destiné à être utilisé par l'application EDD pour implémenter des niveaux d'autorisation et définit comment les attributs METHOD sont présentés.

Structure lexicale

CLASS [ALARM, ANALOG_OUTPUT, COMPUTATION, CONTAINED, CORRECTION, DEVICE, DIAGNOSTIC, DISCRETE, FREQUENCY, HART, INPUT, LOCAL_DISPLAY, OPERATE, OUTPUT, SERVICE, SPECIALIST, TUNE]+

Les attributs CLASS sont spécifiés dans le Tableau 149.

Tableau 149 – Attributs CLASS

Utilisation	Attribut	Description
s	ALARM	METHOD est associé à une alarme (par exemple, spécification des limites d'une alarme, indication de l'état d'alarme, etc.).
s	ANALOG_OUTPUT	METHOD fait partie d'un bloc de sortie analogique.
s	COMPUTATION	METHOD fait partie d'un bloc de calcul.
s	CONTAINED	METHOD représente les caractéristiques physiques de l'appareil.
s	CORRECTION	METHOD prend en charge les transducteurs dans l'appareil. METHOD peut établir des limites de transducteur, fournir un amortissement de signal, indiquer la valeur du transducteur, linéariser ou étalonner le transducteur, etc.
s	DEVICE	METHOD spécifie les caractéristiques physiques de l'appareil.
s	DIAGNOSTIC	METHOD évalue l'état de l'appareil.
s	DISCRETE	METHOD prend en charge l'E/S discrète et/ou numérique de l'appareil.
s	FREQUENCY	METHOD prend en charge l'E/S de fréquence de l'appareil.
s	HART	METHOD fait partie du bloc HART qui caractérise l'interface HART. Il y a un seul bloc HART.
s	INPUT	METHOD fait partie d'un bloc d'entrée. Un bloc d'entrée est un type particulier de bloc de calcul qui effectue des conversions d'unité, une mise à l'échelle et un amortissement. Les paramètres de bloc d'entrée peuvent être déterminés par la sortie d'un autre bloc.
s	LOCAL_DISPLAY	METHOD fait partie du bloc d'affichage local. Un bloc d'affichage local contient les attributs VARIABLE associés à l'interface locale (clavier, affichage, etc.) de l'appareil.
s	OPERATE	METHOD est utilisé pour contrôler l'exploitation d'un bloc.
s	OUTPUT	METHOD fait partie du bloc de sortie. Les valeurs de paramètres de sortie peuvent être accessibles par un autre bloc d'entrée.
s	SERVICE	METHOD est utilisé lors de la conduite de maintenance de routine.
s	SPECIALIST	METHOD doit être uniquement exécutable par un utilisateur spécialiste de l'application EDD. Avec cet attribut CLASS, METHOD ne doit pas être exécutable pour un utilisateur non spécialiste. Si VALIDITY ou VISIBILITY est FALSE, METHOD ne doit pas être visible. Si l'attribut n'est pas défini, METHOD doit être exécutable.
s	TUNE	METHOD est utilisé pour régler l'algorithme d'un bloc.

7.24.2.3 TYPE

Objet

L'attribut TYPE spécifie le type de la valeur retournée.

Structure lexicale

TYPE [char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long, float, double, time_t, DD_STRING]

Les attributs TYPE sont spécifiés dans le Tableau 150.

Tableau 150 – Attributs TYPE

Utilisa-tion	Attribut	Description
s	char	METHOD retourne une valeur de type caractère.
s	unsigned char	METHOD retourne une valeur de type caractère non signé.
s	short	METHOD retourne une valeur de type court.
s	unsigned short	METHOD retourne une valeur de type court non signé.
s	int	METHOD retourne une valeur de type entier.
s	unsigned int	METHOD retourne une valeur de type entier non signé.
s	long	METHOD retourne une valeur de type long.
s	unsigned long	METHOD retourne une valeur de type long non signé.
s	long long	METHOD retourne une valeur de type long long.
s	unsigned long long	METHOD retourne une valeur de type long long non signé.
s	float	METHOD retourne une valeur de type à virgule flottante.
s	double	METHOD retourne une valeur de type à double précision.
s	time_t	METHOD retourne une valeur de type time_t.
s	DD_STRING	METHOD retourne une valeur de type DD_STRING.

7.25 PROGRAM (vide)

7.26 RECORD

Objet

Un attribut RECORD est un groupe logique d'attributs VARIABLE utilisés pour représenter un objet de communication complexe. Chaque attribut MEMBER dans RECORD est une référence à VARIABLE et peut être constitué de différents types de données.

Structure lexicale

RECORD *Identifiant* [MEMBERS, LABEL, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]

Les attributs RECORD sont spécifiés dans le Tableau 151.

Tableau 151 – Attributs RECORD

Utilisa-tion	Attribut	Description
m	MEMBERS	Elément lexical (voir 7.36.12).
m	LABEL	Elément lexical (voir 7.36.9).
o	HELP	Elément lexical (voir 7.36.8).
o	PRIVATE	Elément lexical (voir 7.36.18).
o	RESPONSE_CODES	Elément lexical (voir 7.36.14).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	WRITE_MODE	Elément lexical (voir 7.36.20).

7.27 REFERENCE_ARRAY

7.27.1 Structure générale

Objet

Un attribut REFERENCE_ARRAY est un ensemble d'éléments du type d'élément spécifié (par exemple, VARIABLE ou MENU).

NOTE Les attributs REFERENCE_ARRAY ne sont pas relatifs aux matrices de communication (type d'élément VALUE_ARRAY). Les matrices de communication sont des matrices de valeurs.

Structure lexicale

REFERENCE_ARRAY identifier [item-type, ELEMENTS, HELP, LABEL, PRIVATE, VALIDITY, VISIBILITY]

Les attributs REFERENCE_ARRAY sont spécifiés dans le Tableau 152.

Tableau 152 – Attributs REFERENCE_ARRAY

Utilisa-tion	Attribut	Description
m	item-type	Type d'élément spécifié dans l'attribut ELEMENTS. Doit être un des éléments de base répertoriés dans le Tableau 50.
m	ELEMENTS	Elément lexical (voir 7.27.2).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	PRIVATE	Elément lexical (voir 7.36.18).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).

7.27.2 Attribut spécifique – ELEMENTS

Objet

L'attribut ELEMENTS spécifie une liste d'éléments. Il définit pour chaque élément la référence d'élément, son index associé, sa description et une aide facultatives. Les attributs de description et d'aide, s'ils sont utilisés, remplacent les spécifications correspondantes dans l'EDD elle-même.

Structure lexicale

ELEMENTS [integer, reference, description, help]<exp>]+

Les attributs ELEMENTS sont spécifiés dans le Tableau 153.

Tableau 153 – Attributs ELEMENTS

Utilisa-tion	Attribut	Description
m	integer	Numéro par lequel l'élément peut être référencé. Chaque numéro doit être défini de façon unique dans une construction REFERENCE_ARRAY.
m	reference	Référence à une instance de data-item.
o	description	Brève description pour l'élément.
o	help	Texte d'aide pour l'élément.

7.28 Relations

7.28.1 REFRESH

Objet

L'écriture d'un paramètre sur un appareil peut provoquer la mise à jour par l'appareil des valeurs d'autres paramètres. Une relation REFRESH définit la dépendance entre le jeu cause-parameter et le jeu affected-parameter. Le jeu affected-parameter doit être lu si une valeur du jeu cause-parameter est modifiée, c'est-à-dire définie sur une nouvelle valeur. La modification peut être initiée par des interactions d'un utilisateur dans l'application EDD, dans les exécutions de METHOD ou par la lecture d'une nouvelle valeur depuis l'appareil.

Un attribut VARIABLE peut avoir une relation REFRESH avec lui-même, ce qui signifie que VARIABLE doit être lu après écriture.

L'attribut VARIABLE du jeu affected-parameter peut être utilisé dans une autre relation REFRESH en tant que cause-parameter.

La relation REFRESH ne doit pas être exécutée quand l'application EDD exécute un téléchargement (voir IEC 61804-4).

Structure lexicale

REFRESH identifier, [(cause-parameter) <exp>]+, [(affected-parameter) <exp>]+

Les attributs REFRESH sont spécifiés dans le Tableau 154.

Tableau 154 – Attributs REFRESH

Utilisa-tion	Attribut	Description
m	cause-parameter	Ensemble d'instances de VARIABLE, de RECORD ou de VALUE_ARRAY.
m	affected-parameter	Jeu d'instances de VARIABLE, de LIST, de RECORD ou de VALUE_ARRAY qui doivent être lus, si un des paramètres de cause a été modifié. Un attribut VARIABLE de ce jeu de paramètres affectés peut également être dans le jeu de paramètres de cause d'une autre relation REFRESH.

7.28.2 UNIT

Objet

Une relation UNIT spécifie une référence à VARIABLE contenant une unité et une liste d'éléments dépendants. Lorsque l'attribut VARIABLE contenant l'unité est modifié, la liste d'éléments affectés utilisant cette unité doit être réactualisée. Lorsqu'un élément affecté est affiché, son unité doit également être affichée.

Structure lexicale

UNIT `identifieur`, [`cause-parameter`]`<exp>`], [(`affected-parameter`)]`<exp>`]+

Les attributs UNIT sont spécifiés dans le Tableau 155.

Tableau 155 – Attributs UNIT

Utilisation	Attribut	Description
m	cause-parameter	Référence à une instance de VARIABLE de type énumérée contenant le code d'unité ou à une instance de VARIABLE de type en chaîne contenant la chaîne d'unités.
m	affected-parameter	Liste d'instances de VARIABLE, VALUE_ARRAY, LIST ou AXIS qui utilisent ce code d'unité.

7.28.3 WRITE_AS_ONE

Objet

Une relation WRITE_AS_ONE indique une liste de l'ensemble des éléments de données qui doivent être visibles au moment de l'édition des éléments de données. L'application EDD doit afficher tous les éléments de la relation, même si un seul élément de cette relation est référencé dans un attribut MENU ou EDIT_DISPLAY.

Une relation WRITE_AS_ONE doit uniquement être utilisée lorsqu'un appareil exige que des éléments de données spécifiques soient analysés et modifiés simultanément pour une exploitation correcte. Si les variables peuvent être modifiées séparément, une relation WRITE_AS_ONE ne doit pas être utilisée. Afin de garantir une présentation cohérente, il convient de référencer tous les éléments de données dans l'attribut MENU ou EDIT_DISPLAY.

La relation WRITE_AS_ONE n'affecte que l'interface utilisateur de l'application EDD, et non le comportement de communication. La relation n'impose pas à l'application EDD d'écrire la totalité des éléments si seuls certains de ceux-ci ont été modifiés.

Structure lexicale

WRITE_AS_ONE `identifieur`, [(`reference`)]`<exp>`]+

L'attribut WRITE_AS_ONE est spécifié dans le Tableau 156.

Tableau 156 – Attribut WRITE_AS_ONE

Utilisation	Attribut	Description
m	reference	Liste d'instances de VARIABLE, VALUE_ARRAY et RECORD qui doivent être affichées.

7.29 RESPONSE_CODES

Objet

Les attributs RESPONSE_CODES spécifient les valeurs qu'un appareil peut retourner en tant qu'information d'erreur. Chaque attribut VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST ou COMMAND peut avoir son propre ensemble de RESPONSE_CODES.

Structure lexicale

```
RESPONSE_CODES      identifier,      [(integer,      [DATA_ENTRY_ERROR,
DATA_ENTRY_WARNING,      MISC_ERROR,      MISC_WARNING,      MODE_ERROR,
PROCESS_ERROR, SUCCESS], description, help*)<exp>]+
```

Les attributs RESPONSE_CODES sont spécifiés dans le Tableau 157.

Tableau 157 – Attributs RESPONSE_CODES

Utilisa-tion	Attribut	Description
m	description	Chaîne qui s'affichera lorsque le code de réponse sera retourné par l'appareil.
m	integer	Valeur retournée de l'appareil pour identifier le code de réponse.
s	DATA_ENTRY_ERROR	Le service de couche application a été rejeté parce que les données envoyées ne sont pas valides.
s	DATA_ENTRY_WARNING	Le service de couche application a été accepté et traité avec une version des données envoyées légèrement modifiée.
s	MISC_ERROR	Le service de couche application a été rejeté.
s	MISC_WARNING	Le service de couche application a été accepté et traité comme spécifié et il existe des informations additionnelles, sans rapport avec le service de couche application, qui pourraient intéresser l'utilisateur.
s	MODE_ERROR	Le service de couche application a été rejeté parce que l'appareil était dans un mode dans lequel le service de couche application ne pouvait pas être exécuté.
s	PROCESS_ERROR	Le service de couche application a été rejeté parce qu'un processus appliqué à l'appareil est non valide.
s	SUCCESS	Le service de couche application a été accepté et traité tel que spécifié.
o	help	Texte d'aide pour l'élément.

7.30 SOURCE

7.30.1 Structure générale

Objet

SOURCE décrit une source de valeurs de données pour un attribut CHART.

Structure lexicale

```
SOURCE      identifier      [MEMBERS,      EMPHASIS,      EXIT_ACTIONS,      HELP,
INIT_ACTIONS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY,
VISIBILITY, Y_AXIS]
```

Les attributs SOURCE sont spécifiés dans le Tableau 158.

Tableau 158 – Attributs SOURCE

Utilisa-tion	Attribut	Description
m	MEMBERS	Elément lexical (voir 7.36.12).
o	EMPHASIS	Elément lexical (voir 7.36.5).
o	EXIT_ACTIONS	Elément lexical (voir 7.30.2.1).
o	HELP	Elément lexical (voir 7.36.8).
o	INIT_ACTIONS	Elément lexical (voir 7.30.2.2).
o	LABEL	Elément lexical (voir 7.36.9).
o	LINE_COLOR	Elément lexical (voir 7.36.10).
o	LINE_TYPE	Elément lexical (voir 7.36.11).
o	REFRESH_ACTIONS	Elément lexical (voir 7.30.2.3).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	Y_AXIS	Elément lexical (voir 7.30.2.4).

Pour chaque attribut SOURCE ne spécifiant pas un attribut LINE_TYPE ou LINE_COLOR, chaque ligne doit adopter une apparence différente des autres dans CHART.

7.30.2 Attributs spécifiques

7.30.2.1 EXIT_ACTIONS

Objet

L'attribut EXIT_ACTIONS spécifie les actions qui sont exécutées lorsque l'attribut CHART contenant l'attribut SOURCE est fermé.

Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD restants ne doivent pas être exécutés.

L'attribut EXIT_ACTIONS ne doit pas être exprimé à l'aide d'une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

EXIT_ACTIONS [reference]+, DEFINITION

Les attributs EXIT_ACTIONS sont spécifiés dans le Tableau 188.

7.30.2.2 INIT_ACTIONS

Objet

L'attribut INIT_ACTIONS spécifie des actions qui sont exécutées avant que la valeur de SOURCE ne soit initialement affichée par un attribut CHART.

Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD restants ne doivent pas être exécutés. De plus, l'application EDD ne doit pas représenter graphiquement la valeur de SOURCE si un attribut METHOD se termine pour une raison imprévue.

L'attribut INIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

INIT_ACTIONS [reference]+, DEFINITION

Les attributs INIT_ACTIONS sont spécifiés dans le Tableau 188.

7.30.2.3 REFRESH_ACTIONS

Objet

L'attribut REFRESH_ACTIONS spécifie les actions qui sont exécutées avant l'affichage de la valeur sur l'attribut CHART dans lequel SOURCE est affiché.

Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD restants ne doivent pas être exécutés.

L'attribut REFRESH_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

REFRESH_ACTIONS [reference]+, DEFINITION

Les attributs REFRESH_ACTIONS sont spécifiés dans le Tableau 188.

7.30.2.4 Y_AXIS

Objet

L'attribut Y_AXIS spécifie le mode d'affichage de l'axe Y de l'attribut SOURCE. Chaque attribut SOURCE sur un attribut CHART partage un axe X commun qui est basé sur le temps, mais chaque attribut SOURCE sur un attribut CHART a son propre attribut Y_AXIS.

Structure lexicale

Y_AXIS (reference) <exp>

L'attribut Y_AXIS est spécifié dans le Tableau 159.

Tableau 159 – Attribut Y_AXIS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance d'AXIS.

7.31 TEMPLATE

7.31.1 Structure générale

Objet

TEMPLATE décrit un ensemble de valeurs de paramètre qui est utilisé pour la configuration hors ligne par l'application EDD. Chaque attribut TEMPLATE définit des valeurs par défaut

pour une configuration spécifique à une application. L'attribut `DEFAULT_VALUES` définit les valeurs par défaut des paramètres.

Structure lexicale

TEMPLATE `identifieur` [`DEFAULT_VALUES`, `HELP`, `LABEL`, `VALIDITY`]

Les attributs `TEMPLATE` sont spécifiés dans le Tableau 160.

Tableau 160 – Attributs `TEMPLATE`

Utilisa-tion	Attribut	Description
m	<code>DEFAULT_VALUES</code>	Élément lexical (voir 7.31.2).
m	<code>LABEL</code>	Élément lexical (voir 7.36.9).
o	<code>HELP</code>	Élément lexical (voir 7.36.8).
o	<code>VALIDITY</code>	Élément lexical (voir 7.36.16).

7.31.2 Attribut spécifique – `DEFAULT_VALUES`

Objet

L'attribut `DEFAULT_VALUES` spécifie les réglages par défaut pour les attributs `VARIABLE`, `VALUE_ARRAY` ou `LIST`. Les valeurs à définir doivent être constantes.

Structure lexicale

`DEFAULT_VALUES` [`reference`, (`value*`, (`value+`)*)]+

Les attributs `DEFAULT_VALUES` sont spécifiés dans le Tableau 161.

Tableau 161 – Attributs `DEFAULT_VALUES`

Utilisa-tion	Attribut	Description
m	<code>reference</code>	Référence à une instance de <code>VARIABLE</code> , un membre d'une instance de <code>RECORD</code> , un élément d'une instance de <code>VALUE_ARRAY</code> , une instance de <code>VALUE_ARRAY</code> ou une instance de <code>LIST</code> .
c/m	<code>value</code>	Pour <code>VARIABLE</code> , constante qui spécifie la valeur. Le type dépend du type de données spécifié avec l'attribut de référence. Pour un attribut <code>VALUE_ARRAY</code> ou un attribut <code>LIST</code> , un jeu de valeurs. Peut être imbriqué pour un attribut <code>LIST</code> ou un attribut <code>VALUE_ARRAY</code> multidimensionnel.

Les EDD avec `BLOCK_A` doivent référencer les éléments de données utilisant le référencement par blocs (voir 7.38).

7.32 `VALUE_ARRAY`

7.32.1 Structure générale

Objet

Un attribut `VALUE_ARRAY` est un groupe logique de valeurs. Chaque élément d'un attribut `VALUE_ARRAY` doit être du même type de données et doit correspondre à une construction EDDL. Un élément est référencé depuis un autre emplacement dans l'EDD via l'identificateur et l'index d'élément de `VALUE_ARRAY`.

NOTE Du point de vue de la communication, les éléments individuels de VALUE_ARRAY ne sont pas traités comme des variables individuelles mais simplement comme une valeur groupée.

Structure lexicale

VALUE_ARRAY *identifieur* [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, PRIVATE, RESPONSE_CODES, VALIDITY, VISIBILITY, WRITE_MODE]

Les attributs VALUE_ARRAY sont spécifiés dans le Tableau 162.

Tableau 162 – Attributs VALUE_ARRAY

Utilisa-tion	Attribut	Description
m	LABEL	Elément lexical (voir 7.36.9).
m	NUMBER_OF_ELEMENTS	Elément lexical (voir 7.32.2.1).
m	TYPE	Elément lexical (voir 7.32.2.2).
o	HELP	Elément lexical (voir 7.36.8).
o	PRIVATE	Elément lexical (voir 7.36.18).
o	RESPONSE_CODES	Elément lexical (voir 7.36.14).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	WRITE_MODE	Elément lexical (voir 7.36.20).

7.32.2 Attributs spécifiques

7.32.2.1 NUMBER_OF_ELEMENTS

Objet

L'attribut NUMBER_OF_ELEMENTS spécifie le nombre d'éléments dans l'attribut VALUE_ARRAY (entier supérieur à zéro).

Structure lexicale

NUMBER_OF_ELEMENTS (*integer*, *expression*)

Les attributs NUMBER_OF_ELEMENTS sont spécifiés dans le Tableau 163.

Tableau 163 – Attributs NUMBER_OF_ELEMENTS

Utilisa-tion	Attribut	Description
s	integer	Nombre d'éléments d'un attribut VALUE_ARRAY.
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut NUMBER_OF_ELEMENTS (pour la description des expressions, voir 7.40).

7.32.2.2 TYPE

Objet

L'attribut TYPE est une référence à VARIABLE, COLLECTION, LIST, VALUE_ARRAY ou REFERENCE_ARRAY. Tous les attributs de la construction EDDL s'appliquent à chacun des éléments de VALUE_ARRAY.

Structure lexicale

TYPE reference

L'attribut TYPE est spécifié dans le Tableau 164.

Tableau 164 – Attribut TYPE

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de construction de base d'EDD. La référence doit se terminer sur un élément de données. L'élément référencé dans l'attribut TYPE d'un attribut VALUE_ARRAY ne doit pas contenir d'attribut VALIDITY ou toute autre expression conditionnelle qui provoquerait des modifications de la structure du type des données. Cette restriction s'applique aussi aux éléments contenus.

7.33 VARIABLE

7.33.1 Structure générale

Objet

VARIABLE décrit des paramètres contenus dans un appareil ou dans l'application EDD.

Structure lexicale

VARIABLE identifier, [CLASS, LABEL, TYPE, CONSTANT_UNIT,
DEFAULT_VALUE, HANDLING, HEIGHT, HELP, INITIAL_VALUE,
POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_READ_UPDATE_ACTIONS,
POST_USERCHANGE_ACTIONS , POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS,
PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, PRIVATE, REFRESH_ACTIONS,
RESPONSE_CODES, VALIDITY, VISIBILITY, WIDTH, WRITE_MODE]

Les attributs VARIABLE sont spécifiés dans le Tableau 165.

Tableau 165 – Attributs VARIABLE

Utilisa-tion	Attribut	Description
m	CLASS	Elément lexical (voir 7.33.2.1).
o	LABEL	Elément lexical (voir 7.36.9).
m	TYPE	Elément lexical (voir 7.33.2.2).
o	CONSTANT_UNIT	Elément lexical (voir 7.33.2.3).
o	DEFAULT_VALUE	Elément lexical (voir 7.33.2.4).
o	HANDLING	Elément lexical (voir 7.36.6).
o	HEIGHT	Elément lexical (voir 7.36.7).
o	HELP	Elément lexical (voir 7.36.8).
o	INITIAL_VALUE	Elément lexical (voir 7.33.2.5).
o	POST_EDIT_ACTIONS	Elément lexical (voir 7.33.2.6).
o	POST_READ_ACTIONS	Elément lexical (voir 7.33.2.7).
o	POST_RQSTUPDATE_ACTIONS	Elément lexical (voir 7.33.2.15).
o	POST_USERCHANGE_ACTIONS	Elément lexical (voir 7.33.2.16).
o	POST_WRITE_ACTIONS	Elément lexical (voir 7.33.2.8).
o	PRE_EDIT_ACTIONS	Elément lexical (voir 7.33.2.9).
o	PRE_READ_ACTIONS	Elément lexical (voir 7.33.2.10).
o	PRE_WRITE_ACTIONS	Elément lexical (voir 7.33.2.11).
o	PRIVATE	Elément lexical (voir 7.36.18).
o	REFRESH_ACTIONS	Elément lexical (voir 7.33.2.13).
o	RESPONSE_CODES	Elément lexical (voir 7.36.14).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	WIDTH	Elément lexical (voir 7.36.17).
o	WRITE_MODE	Elément lexical (voir 7.36.20).

7.33.2 Attributs spécifiques

7.33.2.1 CLASS

Objet

L'attribut CLASS spécifie comment la VARIABLE est utilisée par l'appareil et l'application EDD à des fins d'organisation et d'affichage.

Les combinaisons de classes ne sont pas toutes admises. Les restrictions sont définies dans les tableaux de profils (voir Annexe D).

La définition de CLASS est basée sur un modèle dans lequel un appareil est constitué d'une pluralité de blocs fonctionnels différents. Chaque bloc a une entrée et produit une sortie. Des blocs peuvent être combinés de différentes façons pour produire différentes fonctionnalités.

Structure lexicale

```
CLASS [ALARM, ANALOG_INPUT, ANALOG_OUTPUT, COMPUTATION, CONTAINED,
CORRECTION, DEVICE, DIAGNOSTIC, DIGITAL_INPUT, DIGITAL_OUTPUT,
DISCRETE_INPUT, DISCRETE_OUTPUT, DYNAMIC, FREQUENCY_INPUT,
```

FREQUENCY_OUTPUT, HART, INPUT, IS_CONFIG, LOCAL, LOCAL_DISPLAY, OPERATE, OPTIONAL, OUTPUT, SERVICE, SPECIALIST, TEMPORARY, TUNE]+

Les attributs CLASS sont spécifiés dans le Tableau 166.

Tableau 166 – Attributs CLASS

Utilisation	Attribut	Description
s	ALARM	VARIABLE contient des limites d'alarme.
s	ANALOG_INPUT	VARIABLE fait partie d'un bloc d'entrée analogique.
s	ANALOG_OUTPUT	VARIABLE fait partie d'un bloc de sortie analogique.
s	COMPUTATION	VARIABLE fait partie d'un bloc de calcul.
s	CONTAINED	VARIABLE représente les caractéristiques physiques de l'appareil.
s	CORRECTION	VARIABLE fait partie du bloc de correction.
s	DEVICE	VARIABLE représente les caractéristiques physiques de l'appareil.
s	DIAGNOSTIC	VARIABLE indique le statut de l'appareil.
s	DIGITAL_INPUT	VARIABLE fait partie d'un bloc d'entrée numérique.
s	DIGITAL_OUTPUT	VARIABLE fait partie d'un bloc de sortie numérique.
s	DISCRETE_INPUT	VARIABLE fait partie d'un bloc d'entrée discret.
s	DISCRETE_OUTPUT	VARIABLE fait partie d'un bloc de sortie discret.
s	DYNAMIC	VARIABLE est modifiée par l'appareil sans stimulus provenant du réseau.
s	FREQUENCY_INPUT	VARIABLE fait partie d'un bloc d'entrée de fréquence.
s	FREQUENCY_OUTPUT	VARIABLE fait partie d'un bloc de sortie de fréquence.
s	HART	VARIABLE fait partie du bloc HART qui caractérise l'interface HART. Il y a un seul bloc HART.
s	INPUT	VARIABLE fait partie d'un bloc d'entrée. Un bloc d'entrée est un type particulier de bloc de calcul qui effectue des conversions d'unité, une mise à l'échelle et un amortissement. Le paramètre des paramètres de bloc d'entrée peut être déterminé par la sortie d'un autre bloc.
s	IS_CONFIG	Une modification des résultats de VARIABLE lors de la définition du compteur de révision ou du bit modifié par la configuration, comme applicable.
s	LOCAL	Cet attribut VARIABLE est utilisé localement par l'application EDD. Les attributs VARIABLE locaux ne sont pas stockés dans un appareil, mais ils peuvent être envoyés à un appareil.
s	LOCAL_DISPLAY	VARIABLE fait partie du bloc d'affichage local. Un bloc d'affichage local contient les attributs VARIABLE associés à l'interface locale (clavier, affichage, etc.) de l'appareil.
s	OPERATE	VARIABLE est utilisé pour contrôler l'exploitation d'un bloc.
s	OPTIONAL	VARIABLE peut ne pas être présent dans l'appareil.
s	OUTPUT	VARIABLE fait partie du bloc de sortie. Les valeurs de VARIABLE de sortie peuvent être accessibles par un autre bloc d'entrée.
s	SERVICE	VARIABLE est utilisé lors de la conduite de maintenance de routine.
s	SPECIALIST	VARIABLE ne doit être modifiable que si un spécialiste utilise l'application EDD. Pour un utilisateur non spécialiste, VARIABLE avec cet attribut CLASS doit être uniquement lu de manière indépendante de l'attribut HANDLING spécifié. Si VALIDITY ou VISIBILITY est FALSE, VARIABLE ne doit pas être visible. Si l'attribut n'est pas défini, VARIABLE doit être modifiable.
s	TEMPORARY	VARIABLE est initialisé à DEFAULT_VALUE dans chaque session d'application EDD. Les variables LOCAL peuvent être rendues persistantes par l'application EDD; les variables TEMPORARY ne sont jamais persistantes.
s	TUNE	VARIABLE est utilisé pour régler l'algorithme d'un bloc.

7.33.2.2 TYPE

7.33.2.2.1 Structure générale

Objet

L'attribut TYPE décrit le format des valeurs de VARIABLE. Dans le Tableau D.9 la colonne 'valeur initiale' montre les valeurs initiales lorsque l'attribut TYPE est spécifié sans DEFAULT_VALUE et INITIAL_VALUE.

Structure lexicale

TYPE [DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER, DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE, BIT_ENUMERATED, ENUMERATED, INDEX, OBJECT_REFERENCE, ASCII, BITSTRING, EUC, OCTET, PACKED_ASCII, PASSWORD, VISIBLE, BOOLEAN]

Les attributs TYPE sont spécifiés dans le Tableau 167.

Tableau 167 – Attributs TYPE

Utilisa- tion	Attribut	Description
s	BOOLEAN	Type booléen (voir 7.33.2.2.2.8).
s	DOUBLE	Type arithmétique (voir 7.33.2.2.2.1).
s	FLOAT	Type arithmétique (voir 7.33.2.2.2.1).
s	INTEGER	Type arithmétique (voir 7.33.2.2.2.1).
s	UNSIGNED_INTEGER	Type arithmétique (voir 7.33.2.2.2.1).
s	DATE	Type de date/heure (voir 7.33.2.2.2.2).
s	DATE_AND_TIME	Type de date/heure (voir 7.33.2.2.2.2).
s	DURATION	Type de date/heure (voir 7.33.2.2.2.2).
s	TIME	Type de date/heure (voir 7.33.2.2.2.2).
s	TIME_VALUE(8)	Type de date/heure (voir 7.33.2.2.2.2).
s	TIME_VALUE(4)	Type de date/heure (voir 7.33.2.2.2.2).
s	BIT_ENUMERATED	Type d'énumération (voir 7.33.2.2.2.3).
s	ENUMERATED	Type d'énumération (voir 7.33.2.2.2.4).
s	INDEX	Type d'index (voir 7.33.2.2.2.5).
s	OBJECT_REFERENCE	Référence d'objet (voir 7.33.2.2.2.6).
s	ASCII	Type de chaîne (voir 7.33.2.2.2.7).
s	BITSTRING	Type de chaîne (voir 7.33.2.2.2.7).
s	EUC	Type de chaîne (voir 7.33.2.2.2.7).
s	OCTET	Type de chaîne (voir 7.33.2.2.2.7).
s	PACKED_ASCII	Type de chaîne (voir 7.33.2.2.2.7).
s	PASSWORD	Type de chaîne (voir 7.33.2.2.2.7).
s	VISIBLE	Type de chaîne (voir 7.33.2.2.2.7).

7.33.2.2.2 Attributs spécifiques

7.33.2.2.2.1 DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Objet

Les attributs VARIABLE de TYPE FLOAT et DOUBLE sont respectivement des nombres à virgule flottante au format de base simple précision et au format de base double précision, tel que défini dans l'IEEE 754. Les attributs VARIABLE de TYPE INTEGER et UNSIGNED_INTEGER sont respectivement des nombres entiers signés et non signés.

Ces types sont appelés collectivement types arithmétiques.

Structure lexicale — DOUBLE

```
DOUBLE [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE,
[MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR], [(description,
help, value)<exp>]*
```

Structure lexicale — FLOAT

```
FLOAT [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE,
[MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR], [(description,
help, value)<exp>]*
```

Structure lexicale — INTEGER

```
INTEGER [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE,
[MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR, size], [(description,
help, value)*
```

Structure lexicale — UNSIGNED_INTEGER

```
UNSIGNED_INTEGER [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT,
INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*, SCALING_FACTOR,
size], [(description, help, value)<exp>]*
```

Les attributs des types arithmétiques sont spécifiés dans le Tableau 168.

Tableau 168 – Attributs DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Utilisa-tion	Attribut	Description
o	DEFAULT_VALUE	Spécifie la valeur par défaut de VARIABLE. DEFAULT_VALUE est disponible pour tous les types EDDL. Si aucun INITIAL_VALUE n'est défini, DEFAULT_VALUE est utilisé en tant qu'INITIAL_VALUE. DEFAULT_VALUE peut également être généré par une expression (voir 7.40). Structure lexicale DEFAULT_VALUE (double)<exp> DEFAULT_VALUE (float)<exp> DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> D EFAULT_VALUE (expression)<exp>
o	description	Brève description de l'élément.
o	DISPLAY_FORMAT	Spécifie comment la valeur de l'attribut VARIABLE est affichée. La chaîne DISPLAY_FORMAT doit être définie conformément aux fonctions printf et scanf du langage de programmation C (voir également ISO/IEC 9899). Structure lexicale DISPLAY_FORMAT (string)<exp>

Utilisa-tion	Attribut	Description
o	EDIT_FORMAT	Spécifie le format d'édition. La chaîne EDIT_FORMAT doit être définie conformément aux fonctions printf et scanf du langage de programmation C (voir également ISO/IEC 9899). Structure lexicale EDIT_FORMAT (string)<exp>
o	help	Spécifie le texte d'aide pour l'élément.
o	INITIAL_VALUE	Lorsqu'un appareil est instancié pour la première fois, l'attribut INITIAL_VALUE d'un attribut VARIABLE est affiché. Il est fréquent que DEFAULT_VALUE soit conditionné et dépende d'autres paramètres. Dans ce cas, l'attribut INITIAL_VALUE est utilisé pour reconstituer les paramètres par défaut. INITIAL_VALUE est disponible pour tous les types et ne doit pas être conditionné. Structure lexicale INITIAL_VALUE (double) INITIAL_VALUE (float) INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)
o	MAX_VALUE	Spécifie la limite supérieure de valeurs auxquelles VARIABLE doit être réglé. Si l'attribut VARIABLE est CLASS DYNAMIC, il convient de régler l'appareil à une valeur de VARIABLE comprise dans l'intervalle spécifié par ses valeurs minimale et maximale. Un attribut VARIABLE peut avoir plus d'une valeur maximale, par exemple, pour spécifier plus d'un intervalle de validité. Lorsqu'il y a plusieurs valeurs maximales, un entier additionnel est utilisé pour numéroter chaque attribut MAX_VALUE. Chaque attribut MAX_VALUE doit avoir un attribut MIN_VALUE équivalent. L'intervalle spécifié avec une paire de MIN_VALUE et MAX_VALUE peut ne pas chevaucher un autre intervalle spécifié. Si un attribut MIN_VALUE/MAX_VALUE n'a pas d'attribut MAX_VALUE/MIN_VALUE associé, la limite supérieure/inférieure est non limitée et peut ne pas être spécifiée. L'attribut MAX_VALUE peut également être généré par une expression (voir 7.40). Structure lexicale MAX_VALUE (double, integer)<exp> MAX_VALUE (float, integer)<exp> MAX_VALUE (integer, integer)<exp> MAX_VALUE (unsigned_integer, integer)<exp> MAX_VALUE (expression)<exp>
c	m	Entier croissant commençant à 1, qui identifie l'attribut MAX_VALUE s'il y en plus d'un.
o	MIN_VALUE	Spécifie la limite inférieure de valeurs auxquelles VARIABLE doit être réglé. Si l'attribut VARIABLE est CLASS DYNAMIC, il convient de régler l'appareil à une valeur de VARIABLE comprise dans l'intervalle spécifié par ses valeurs minimale et maximale. Un attribut VARIABLE peut avoir plus d'une valeur minimale, par exemple, pour spécifier plus d'un intervalle de validité. Lorsqu'il existe plusieurs valeurs minimales, un entier additionnel est utilisé pour numéroter chaque attribut MIN_VALUE. Chaque attribut MIN_VALUE doit avoir un attribut MAX_VALUE équivalent. L'intervalle spécifié avec une paire de MIN_VALUE et MAX_VALUE peut ne pas chevaucher un autre intervalle spécifié. Si un attribut MIN_VALUE/MAX_VALUE n'a pas d'attribut MAX_VALUE/MIN_VALUE associé, la limite supérieure/inférieure est non limitée et peut ne pas être spécifiée. L'attribut MIN_VALUE peut également être construit par une expression (voir 7.40). Structure lexicale MIN_VALUE (double, integer)<exp> MIN_VALUE (float, integer)<exp> MIN_VALUE (integer, integer)<exp> MIN_VALUE (unsigned_integer, integer)<exp> MIN_VALUE (expression)<exp>

Utilisa-tion	Attribut	Description
c	n	Entier croissant commençant à 1, qui identifie l'attribut MIN_VALUE s'il y en a plus d'un.
o	SCALING_FACTOR	Indique que la valeur affichée de VARIABLE n'est pas la valeur retournée par un appareil. La valeur affichée est la valeur retournée par un appareil multipliée par un facteur. Par conséquent, le processeur EDDL doit multiplier la valeur de VARIABLE retournée par un appareil par SCALING_FACTOR avant qu'elle ne soit affichée (ou avant qu'elle ne soit utilisée autrement). L'attribut SCALING_FACTOR ne doit pas être égal à zéro. L'attribut SCALING_FACTOR peut également être généré par une expression (voir 7.40). L'attribut SCALING_FACTOR est appliqué pour l'affichage uniquement; la valeur de la variable, lorsqu'elle est assignée à une variable locale de méthode, est la valeur non mise à l'échelle retournée par l'appareil; les valeurs minimales et maximales, lorsqu'elles sont utilisées conjointement avec le facteur d'échelle, sont également des valeurs non mises à l'échelle. Structure lexicale SCALING_FACTOR (double)<exp> SCALING_FACTOR (expression)<exp>
o	size	Taille du type de données en octets. La taille est une constante entière supérieure à zéro et n'a pas de limite supérieure. Cette valeur est facultative. La valeur par défaut est 1.
o	value	Constante qui spécifie la valeur. Le type dépend du type de données spécifié avec l'attribut TYPE qui est facultatif, mais s'il est défini, un attribut de description est exigé. Des valeurs égales ne sont pas admises.

7.33.2.2.2.2 DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

Objet

Les attributs VARIABLE de type de données DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE sont utilisés pour spécifier des dates, heures, dates et heures, et des différences d'heure:

- le type de données DATE est constitué d'une date calendaire;
- le type de données DATE_AND_TIME (date/heure) est constitué d'une date calendaire et d'une heure;
- le type de données DURATION est une différence d'heure qui est constituée d'un temps en ms et d'un compteur de jours;
- le type de données TIME est constitué d'une heure et d'une date facultative;
- le type de données TIME_VALUE est utilisé pour représenter des différences d'heure, l'heure du jour ou les valeurs de temps absolues avec la précision exigée pour la synchronisation d'horloge d'application.

Structure lexicale – DATE

DATE spécifie que VARIABLE est un entier codé sur trois octets. Le codage des octets est spécifié en D.6.2.2.

DATE [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Structure lexicale – DATE_AND_TIME

DATE_AND_TIME spécifie que VARIABLE est un entier codé sur sept octets. Le codage des octets est spécifié en D.6.2.3.

DATE_AND_TIME [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Structure lexicale – DURATION

DURATION spécifie que VARIABLE est un entier codé sur 6 octets. Les quatre premiers bits doivent avoir la valeur zéro. Les bits 0 à 27 représentent le temps en ms. Le nombre de jours est une valeur binaire de 16 bits. Le codage des octets est spécifié en D.6.2.4.

DURATION [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Structure lexicale – TIME

TIME spécifie que VARIABLE est un entier codé sur six octets. L'heure est exprimée en ms à partir de minuit. A minuit, le compteur recommence à zéro.

La date est indiquée en jours par rapport au 01/01/1984. Le premier jour de janvier 1984, la date commence à la valeur zéro. Le codage des octets est spécifié en D.6.2.5.

TIME [DEFAULT_VALUE, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*]

Structure lexicale – TIME_VALUE

TIME_VALUE spécifie que VARIABLE est un entier codé sur quatre à huit octets. Ce nombre à décimale fixe représente le temps exprimé en multiples de 1/32 ms. Le codage des octets est spécifié en D.6.2.6.

TIME_VALUE [DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, [MAX_VALUE, m]*, [MIN_VALUE, n]*, size, TIME_FORMAT, TIME_SCALE]

Les attributs des types DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE sont spécifiés dans le Tableau 169.

Tableau 169 – Attributs DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

Utilisa-tion	Attribut	Description
o	DEFAULT_VALUE	Spécifie la valeur par défaut de VARIABLE. L'attribut DEFAULT_VALUE est disponible pour tous les types EDDL. Si aucun attribut INITIAL_VALUE n'est défini, l'attribut DEFAULT_VALUE est utilisé en tant qu'attribut INITIAL_VALUE. L'attribut DEFAULT_VALUE peut également être généré par une expression (voir 7.40). Structure lexicale DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	DISPLAY_FORMAT	Voir Tableau 168.
o	EDIT_FORMAT	Voir Tableau 168.
o	INITIAL_VALUE	Lorsqu'un appareil est instancié pour la première fois, l'attribut INITIAL_VALUE d'un attribut VARIABLE est affiché. Il est fréquent que l'attribut DEFAULT_VALUE soit conditionné et dépende d'autres paramètres. Dans ce cas, l'attribut INITIAL_VALUE est utilisé pour reconstituer les paramètres par défaut. INITIAL_VALUE est disponible pour tous les types et ne doit pas être conditionné. Structure lexicale INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)

Utilisa-tion	Attribut	Description
o	MAX_VALUE	Spécifie la limite supérieure de valeurs auxquelles VARIABLE doit être réglé. Si l'attribut VARIABLE est CLASS DYNAMIC, il convient de régler l'appareil à une valeur de VARIABLE comprise dans l'intervalle spécifié par ses valeurs minimale et maximale. Un attribut VARIABLE peut avoir plus d'une valeur maximale, par exemple, pour spécifier plus d'un intervalle de validité. Lorsqu'il y a plusieurs valeurs maximales, un entier additionnel est utilisé pour numéroté chaque attribut MAX_VALUE. Chaque attribut MAX_VALUE doit avoir un attribut MIN_VALUE équivalent. L'intervalle spécifié avec une paire de MIN_VALUE et MAX_VALUE peut ne pas chevaucher un autre intervalle spécifié. Si un attribut MIN_VALUE/MAX_VALUE n'a pas de MAX_VALUE/MIN_VALUE associé, la limite supérieure/inférieure est non limitée et peut ne pas être spécifiée. L'attribut MAX_VALUE peut également être généré par une expression (voir 7.40). Structure lexicale MAX_VALUE (date)<exp> MAX_VALUE (date_and_time)<exp> MAX_VALUE (duration)<exp> MAX_VALUE (time)<exp> MAX_VALUE (time_value)<exp> MAX_VALUE (expression)<exp>
c	m	Entier croissant commençant à 1, qui identifie l'attribut MAX_VALUE s'il y en plus d'un.
o	MIN_VALUE	Spécifie la limite inférieure de valeurs auxquelles VARIABLE doit être réglé. Si l'attribut VARIABLE est CLASS DYNAMIC, il convient de régler l'appareil à une valeur de VARIABLE comprise dans l'intervalle spécifié par ses valeurs minimale et maximale. Un attribut VARIABLE peut avoir plus d'une valeur minimale, par exemple, pour spécifier plus d'un intervalle de validité. Lorsqu'il existe plusieurs valeurs minimales, un entier additionnel est utilisé pour numéroté chaque attribut MIN_VALUE. Chaque attribut MIN_VALUE doit avoir un attribut MAX_VALUE équivalent. L'intervalle spécifié avec une paire de MIN_VALUE et MAX_VALUE peut ne pas chevaucher un autre intervalle spécifié. Si un attribut MIN_VALUE/MAX_VALUE n'a pas de MAX_VALUE/MIN_VALUE associé, la limite supérieure/inférieure est non limitée et peut ne pas être spécifiée. L'attribut MIN_VALUE peut également être construit par une expression (voir 7.40). Structure lexicale MIN_VALUE (date)<exp> MIN_VALUE (date_and_time)<exp> MIN_VALUE (duration)<exp> MIN_VALUE (time)<exp> MIN_VALUE (time_value)<exp> MIN_VALUE (expression)<exp>
c	n	Entier croissant commençant à 1, qui identifie l'attribut MIN_VALUE s'il y en a plus d'un.
o	size	Taille du type de données en octets. Doit être de 4 ou 8. La valeur par défaut est 4. Cet attribut est applicable uniquement à TIME_VALUE.
o	TIME_FORMAT	Spécifie comment l'attribut VARIABLE est affiché et édité. Les chaînes TIME_FORMAT peuvent comprendre les spécificateurs de format %H, %I, %M, %p, %R, %S, et %T DE la fonction strftime du langage de programmation C. Pour DATE_AND_TIME et TIME_VALUE(8), les spécificateurs de format de temps supplémentaires suivants s'appliquent: %a %A %b %B %c %C %d %D %e %F %g %G %h %j %m %u %U %V %w %W %x %y %Y. Lorsque TIME_FORMAT est spécifié, DISPLAY_FORMAT et EDIT_FORMAT ne doivent pas être spécifiés. Structure lexicale TIME_FORMAT (string)<exp>

Utilisa-tion	Attribut	Description
o	TIME_SCALE	Spécifie comment VARIABLE est mis à l'échelle, ce qui peut correspondre à: SECONDS (secondes), MINUTES (minutes) ou HOURS (heures). Lorsqu'il est présent, TIME_SCALE déduit un facteur d'échelle de 1/32 000, 1/1 920 000 ou 1/115 200 000 et une unité constante de secondes (s), minutes (min) ou heures (h), respectivement. Lorsque TIME_SCALE est spécifié, DISPLAY_FORMAT et EDIT_FORMAT peuvent également être spécifiés. Cet attribut est applicable uniquement à TIME_VALUE(4). Structure lexicale TIME_SCALE (SECONDS, MINUTES, HOURS)<exp>

7.33.2.2.2.3 BIT_ENUMERATED

Objet

Un attribut VARIABLE de type BIT_ENUMERATED est un entier non signé défini comme un ensemble de définitions de bit unique. Chaque bit est identifié par la valeur de sa position de bit. Le bit 0 est identifié par la valeur 1 (2^0), le bit 1 par la valeur 2 (2^1), le bit 2 par la valeur 4 (2^2), le bit 3 par la valeur 8 (2^3), etc.

Structure lexicale

```
BIT_ENUMERATED [(description, value, action, function, help, status-
class*)<exp>]+, [DEFAULT_VALUE, INITIAL_VALUE, size]
```

Les attributs BIT_ENUMERATED sont spécifiés dans le Tableau 170.

Tableau 171 – Attributs status-class

Utilisa-tion	Attribut	Description
m	DATA	Configuration de données non valide.
m	HARDWARE	Défaillance du matériel.
m	MISC	Conditions diverses.
m	MODE	L'appareil est dans un mode particulier.
m	PROCESS	Problème avec le processus connecté à l'appareil.
m	SOFTWARE	Défaillance du logiciel.
m	CORRECTABLE	Le bit peut être effacé par l'application EDD ou le processus connecté.
m	SELF_CORRECTING	Le bit sera réinitialisé sans intervention supplémentaire.
m	UNCORRECTABLE	Le bit n'est ni auto-correctible, ni corrigible.
m	EVENT	Événement unique.
m	STATE	L'appareil est dans un état particulier.
m	COMM_ERROR	Défaillance de la communication dans l'autre appareil.
o	IGNORE_IN_HANDHELD	Il convient qu'un bit soit ignoré dans des appareils de communication portatifs.
o	IGNORE_IN_TEMPORARY_MASTER	Il convient qu'un bit soit ignoré dans des appareils maîtres temporaires.
m	MORE	Des informations de statut additionnelles sont disponibles en provenance de l'appareil.
o	ALL	Affecte toutes les sorties (voir Tableau 172).
o	AO	Affecte une des sorties analogiques (voir Tableau 172).
m	BAD	La sortie n'est pas fiable, il convient de ne pas l'utiliser pour la commande (voir Tableau 172).
o	DV	Affecte une des variables dynamiques (voir Tableau 172).
o	TV	Affecte une des variables d'émetteur (voir Tableau 172).
m	DETAIL	Le bit est répété ailleurs dans un bit de classe SUMMARY.
m	SUMMARY	Le bit est une combinaison logique d'autres bits de classe DETAIL.
o	INFO	Le bit donne des informations sur le statut du processus ou de l'appareil.
o	WARNING	Le bit donne un avertissement sur le statut du processus ou de l'appareil.
o	ERROR	Le bit donne une erreur sur le statut du processus ou de l'appareil.
o	IGNORE_IN_HOST	Le bit doit être ignoré dans un hôte fixe, raccordé en permanence.

Les status-classes ALL, AO, DV et TV donnent des informations de statut supplémentaires sur la sortie d'un appareil.

Structure lexicale — ALL

ALL [AUTO_BAD, AUTO_GOOD, MANUAL_BAD, MANUAL_GOOD]

Structure lexicale — AO

AO [AUTO_BAD, AUTO_GOOD, integer, MANUAL_BAD, MANUAL_GOOD]

Structure lexicale — DV

DV [AUTO_BAD, AUTO_GOOD, integer, MANUAL_BAD, MANUAL_GOOD]

Structure lexicale — TV

TV [AUTO_BAD, AUTO_GOOD, integer, MANUAL_BAD, MANUAL_GOOD]

Les attributs sont spécifiés dans le Tableau 172.

Tableau 172 – Attributs ALL, AO, DV, TV

Utilisa-tion	Attribut	Description
s	AUTO_BAD	La valeur provient d'un processus d'application et n'est pas valide.
s	AUTO_GOOD	La valeur provient d'un processus d'application et n'est pas valide.
m	integer	La valeur de VARIABLE provient de l'appareil.
s	MANUAL_BAD	La valeur ne provient pas d'un processus d'application et n'est pas valide.
s	MANUAL_GOOD	La valeur ne provient pas d'un processus d'application et n'est pas valide.

7.33.2.2.2.4 ENUMERATED**Objet**

Un attribut VARIABLE de type ENUMERATED est un entier non signé défini comme un ensemble de valeurs énumérées prédéfinies.

Structure lexicale

ENUMERATED [(description, value, help)<exp>]*, [DEFAULT_VALUE, INITIAL_VALUE, size]

Les attributs de type ENUMERATED sont spécifiés dans le Tableau 173.

Tableau 173 – Attributs de type ENUMERATED

Utilisa-tion	Attribut	Description
m	description	Voir Tableau 168.
m	value	Constante qui spécifie la valeur de VARIABLE. Le type est un entier non signé. La liste d'énumération est facultative, mais si elle est définie, une valeur et une description sont exigées. Des valeurs égales ne sont pas admises.
o	DEFAULT_VALUE	Spécifie la valeur par défaut de VARIABLE. DEFAULT_VALUE est disponible pour tous les types EDDL. Si aucun INITIAL_VALUE n'est défini, DEFAULT_VALUE est utilisé en tant qu'INITIAL_VALUE. DEFAULT_VALUE peut également être généré par une expression (voir 7.40). Structure lexicale DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	help	Voir Tableau 168.
o	INITIAL_VALUE	Lorsqu'un appareil est instancié pour la première fois, l'attribut INITIAL_VALUE d'un attribut VARIABLE est affiché. Il est fréquent que l'attribut DEFAULT_VALUE soit conditionné et dépende d'autres paramètres. Dans ce cas, l'attribut INITIAL_VALUE est utilisé pour reconstituer les paramètres par défaut. INITIAL_VALUE est disponible pour tous les types et ne doit pas être conditionné. Structure lexicale INITIAL_VALUE (unsigned_integer)
o	size	Voir Tableau 168.

7.33.2.2.2.5 INDEX

Objet

Un attribut VARIABLE de type INDEX est un entier non signé qui est utilisé pour référencer les éléments de REFERENCE_ARRAY (voir 7.27). L'attribut entier ELEMENTS (voir 7.27.2) spécifie les valeurs admises de VARIABLE.

Lorsqu'un attribut VARIABLE INDEX est présenté à l'utilisateur, le texte associé aux index de la matrice doit être affiché, mais pas la valeur numérique de VARIABLE.

Structure lexicale

INDEX (reference, size)<exp>

Les attributs de type INDEX sont spécifiés dans le Tableau 174.

Tableau 174 – Attributs de type INDEX

Utilisa-tion	Attribut	Description
o	size	Taille de VARIABLE, en octets. Cette valeur est une constante entière supérieure à zéro et n'a pas de limite supérieure. La valeur par défaut est 1. L'attribut REFERENCE_ARRAY doit contenir uniquement des éléments qui ne dépassent pas la taille d'index de la variable.
m	reference	Référence à une instance de REFERENCE_ARRAY.

7.33.2.2.2.6 OBJECT_REFERENCE

Objet

Les attributs VARIABLE du type OBJECT_REFERENCE permettent d'accéder à des éléments d'autres instances de COMPONENT. Des attributs OBJECT_REFERENCE peuvent être utilisés pour naviguer dans la hiérarchie d'un objet, par exemple pour un appareil modulaire (voir IEC 61804-4).

Structure lexicale

OBJECT_REFERENCE

7.33.2.2.2.7 ASCII, BITSTRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE

Objet

Les variables de chaîne comprennent les types suivants:

- ASCII – utilisé pour spécifier une séquence de caractères du jeu de caractères ISO Latin-1.
- BITSTRING – utilisé pour spécifier une séquence de bits.
- EUC (unicode étendu) – est le code interne pour des caractères spécifiques (par exemple, Chine: EUC-CN, Taïwan: EUC-TW). EUC est utilisé pour gérer les langues asiatiques (ISO/IEC 10646-1).
- PACKED_ASCII – est un sous-ensemble d'ASCII obtenu en enlevant les deux bits les plus significatifs de chaque caractère ASCII (voir D.6.2.7).

- **PASSWORD** – est un type chaîne prévu pour spécifier des chaînes de mots de passe. A l'exception du mode de présentation de la variable à l'utilisateur, les types de chaînes de mot de passe et ASCII sont identiques.
- **VISIBLE** – cette chaîne est une séquence ordonnée de caractères du jeu de caractères de l'ISO/IEC 10646-1 et l'ISO 2375.
- **OCTET** – est utilisé pour spécifier une séquence de données binaires non formatées dont la définition est déterminée par l'implémentation de l'appareil.

Structure lexicale — EUC

EUC [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Structure lexicale — ASCII

ASCII [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Structure lexicale — PACKED_ASCII

PACKED_ASCII [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Structure lexicale — PASSWORD

PASSWORD [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Structure lexicale — BITSTRING

BITSTRING [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Structure lexicale — VISIBLE

VISIBLE [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Structure lexicale — OCTET

OCTET [(string, description, help)<exp>]+, [size, DEFAULT_VALUE, INITIAL_VALUE]

Les attributs des types de chaînes sont spécifiés dans le Tableau 175.

Tableau 175 – Attributs de types de chaînes

Utilisa-tion	Attribut	Description
m	size	Taille du type de données en octets ou pour le nombre de bits de BITSTRING. La taille est une constante entière supérieure à zéro et n'a pas de limite supérieure. Cette valeur n'est pas facultative; une taille doit être spécifiée.
o	DEFAULT_VALUE	Spécifie la valeur par défaut de VARIABLE. L'attribut DEFAULT_VALUE est disponible pour tous les types EDDL. Si aucun attribut INITIAL_VALUE n'est défini, l'attribut DEFAULT_VALUE est utilisé en tant qu'attribut INITIAL_VALUE. Structure lexicale DEFAULT_VALUE (string)<exp>
o	INITIAL_VALUE	Lorsqu'un appareil est instancié pour la première fois, l'attribut INITIAL_VALUE d'un attribut VARIABLE est affiché. Il est fréquent que l'attribut DEFAULT_VALUE soit conditionné et dépende d'autres paramètres. Dans ce cas, l'attribut INITIAL_VALUE est utilisé pour reconstituer les paramètres par défaut. INITIAL_VALUE est disponible pour tous les types et ne doit pas être conditionné. Structure lexicale INITIAL_VALUE (string)
o	string	Constante de chaîne. La liste d'énumération est facultative, mais si elle est définie, une chaîne et une description sont exigées. Des chaînes égales ne sont pas admises.
o	description	Voir Tableau 168.
o	help	Voir Tableau 168.

7.33.2.2.2.8 BOOLEAN**Objet**

Un attribut VARIABLE de type BOOLEAN est un entier non signé sur un bit avec la valeur TRUE ou FALSE. La valeur 0 reflète FALSE, toute autre valeur reflète TRUE.

Structure lexicale

BOOLEAN

7.33.2.3 CONSTANT_UNIT**Objet**

L'attribut CONSTANT_UNIT est utilisé si VARIABLE a un code d'unité qui ne change jamais. L'attribut CONSTANT_UNIT est spécifié comme étant une chaîne qui sera affichée avec la valeur. Un attribut VARIABLE sans unité constante ne comporte pas d'unité associée à celle-ci ou bien les unités ne sont pas constantes.

Structure lexicale

CONSTANT_UNIT (string)<exp>

L'attribut CONSTANT_UNIT est spécifié dans le Tableau 176.

Tableau 176 – Attribut CONSTANT_UNIT

Utilisa-tion	Attribut	Description
m	String	Chaîne de code d'unités à afficher.

7.33.2.4 DEFAULT_VALUE**Objet**

L'attribut DEFAULT_VALUE spécifie le réglage par défaut de VARIABLE. Si aucun attribut INITIAL_VALUE n'est défini, l'attribut DEFAULT_VALUE est utilisé en tant qu'attribut INITIAL_VALUE.

Structure lexicale

DEFAULT_VALUE (expression)<exp>

Le type de l'expression doit être cohérent avec le type de VARIABLE. L'attribut DEFAULT_VALUE peut également être spécifié via l'attribut TYPE de VARIABLE.

L'attribut DEFAULT_VALUE est spécifié dans le Tableau 177.

Tableau 177 – Attribut DEFAULT_VALUE

Utilisa-tion	Attribut	Description
m	expression	La valeur par défaut dépend du type de la variable: - DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER: voir Tableau 168; - DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE: voir Tableau 169; - BIT_ENUMERATED: voir Tableau 170; - ENUMERATED: voir Tableau 173; - ASCII, BITSTRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE: voir Tableau 175.

7.33.2.5 INITIAL_VALUE**Objet**

L'attribut INITIAL_VALUE spécifie la valeur d'un attribut VARIABLE qui est affiché lorsqu'un appareil est instancié pour la première fois. Il est fréquent que l'attribut DEFAULT_VALUE soit conditionné et dépende d'autres paramètres. Dans ce cas, l'attribut INITIAL_VALUE est utilisé pour reconstituer les paramètres par défaut. INITIAL_VALUE est disponible pour tous les types et ne doit pas être conditionné.

Structure lexicale

Expression d'INITIAL_VALUE

Le type de l'expression doit être cohérent avec le type de VARIABLE. L'attribut INITIAL_VALUE peut également être spécifié via l'attribut TYPE de la VARIABLE.

L'attribut INITIAL_VALUE est spécifié dans le Tableau 178.

Tableau 178 – Attribut INITIAL_VALUE

Utilisa-tion	Attribut	Description
m	expression	La valeur initiale dépend du type de la variable: - DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER: voir Tableau 168; - DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE: voir Tableau 169; - BIT_ENUMERATED: voir Tableau 170; - ENUMERATED: voir Tableau 173; - ASCII, BITSTRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE: voir Tableau 175.

7.33.2.6 POST_EDIT_ACTIONS**Objet**

L'attribut POST_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après que l'utilisateur a terminé de modifier l'attribut VARIABLE. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

L'attribut POST_EDIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

POST_EDIT_ACTIONS [reference]*, DEFINITION

Les attributs POST_EDIT_ACTIONS sont spécifiés dans le Tableau 179.

Tableau 179 – Attributs POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS

Utilisa-tion	Attribut	Description
o	reference	Référence à une instance de METHOD ou à une instance de METHOD avec arguments.
o	DEFINITION	Elément lexical (voir 7.36.4).

7.33.2.7 POST_READ_ACTIONS**Objet**

L'attribut POST_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après la lecture de l'attribut VARIABLE sur l'appareil, à savoir après réception de l'accusé de lecture et doivent exploiter les données reçues. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Si l'un des attributs METHOD de PRE_READ_ACTIONS est annulé, l'attribut VARIABLE ne doit pas être lu sur l'appareil et les attributs METHOD de POST_READ_ACTIONS ne doivent pas être appliqués.

L'attribut POST_READ_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

POST_READ_ACTIONS [reference]+, DEFINITION

Les attributs POST_READ_ACTIONS sont spécifiés dans le Tableau 179.

7.33.2.8 POST_WRITE_ACTIONS

Objet

L'attribut POST_WRITE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après l'écriture de l'attribut VARIABLE sur l'appareil, à savoir après réception de la réponse d'écriture et doivent exploiter les données reçues, le cas échéant. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Si l'un des attributs METHOD de PRE_WRITE_ACTIONS est annulé, l'attribut VARIABLE ne doit pas être écrit sur l'appareil et les attributs METHOD de POST_WRITE_ACTIONS ne doivent pas être appliqués.

L'attribut POST_WRITE_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

POST_WRITE_ACTIONS [reference]+, DEFINITION

Les attributs POST_WRITE_ACTIONS sont spécifiés dans le Tableau 179.

7.33.2.9 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement lorsque VARIABLE va être modifié. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

L'attribut PRE_EDIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

PRE_EDIT_ACTIONS [reference]+, DEFINITION

Les attributs PRE_EDIT_ACTIONS sont spécifiés dans le Tableau 179.

7.33.2.10 PRE_READ_ACTIONS

Objet

L'attribut PRE_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant que VARIABLE ne soit lu. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

L'attribut PRE_READ_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

PRE_READ_ACTIONS [reference]+, DEFINITION

Les attributs PRE_READ_ACTIONS sont spécifiés dans le Tableau 179.

7.33.2.11 PRE_WRITE_ACTIONS

Objet

L'attribut PRE_WRITE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant que VARIABLE ne soit écrite dans l'appareil. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

L'attribut PRE_WRITE_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

PRE_WRITE_ACTIONS [reference]+, DEFINITION

Les attributs PRE_WRITE_ACTIONS sont spécifiés dans le Tableau 179.

7.33.2.12 READ_TIMEOUT, WRITE_TIMEOUT (vide)

7.33.2.13 REFRESH_ACTIONS

Objet

L'attribut REFRESH_ACTIONS spécifie les attributs METHOD qui doivent être exécutés lorsque la valeur de VARIABLE est demandée. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

L'attribut REFRESH_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

REFRESH_ACTIONS [reference]+, DEFINITION

Les attributs REFRESH_ACTIONS sont spécifiés dans le Tableau 179.

7.33.2.14 STYLE (vide)

L'attribut STYLE est déconseillé. Les attributs HEIGHT et WIDTH le remplacent (voir 7.36.7 et 7.36.17).

7.33.2.15 POST_RQSTUPDATE_ACTIONS

Objet

POST_RQSTUPDATE_ACTIONS ne doit être pris en charge que par les simulateurs d'appareil.

L'attribut PREPOST_RQSTUPDATE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés quand la valeur de VARIABLE est écrite dans le simulateur d'appareil. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Structure lexicale

POST_RQSTCHANGE_ACTIONS [(reference)<exp>]+

L'attribut POST_RQSTCHANGE_ACTIONS est spécifié dans le Tableau 180.

Tableau 180 – Attribut POST_USERCHANGE_ACTIONS, POST_RQSTUPDATE_ACTIONS

Utilisa-tion	Attribut	Description
m	reference	Référence à un attribut METHOD.

7.33.2.16 POST_USERCHANGE_ACTIONS

Objet

POST_USERCHANGE_ACTIONS ne doit être pris en charge que par les simulateurs d'appareil.

L'attribut POST_USERCHANGE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés quand la modification de VARIABLE dans l'application de simulation d'appareil est terminée. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés.

Structure lexicale

POST_USERCHANGE_ACTIONS [(reference)<exp>]+

L'attribut POST_USERCHANGE_ACTIONS est spécifié dans le Tableau 180.

7.34 VARIABLE_LIST

Objet

VARIABLE_LIST est un groupe d'objets de communication d'EDD (VARIABLE, VALUE_ARRAY ou RECORDS). Les attributs VARIABLE_LIST sont utilisés pour grouper des objets afin de faciliter l'exécution des applications.

VARIABLE_LIST peut être transféré sous la forme d'un objet de communication.

Structure lexicale

VARIABLE_LIST identifier, [MEMBERS, HELP, LABEL, RESPONSE_CODES]

Les attributs VARIABLE_LIST sont spécifiés dans le Tableau 181.

Tableau 181 – Attributs VARIABLE_LIST

Utilisa-tion	Attribut	Description
m	MEMBERS	Elément lexical (voir 7.36.12).
o	HELP	Elément lexical (voir 7.36.8).
o	LABEL	Elément lexical (voir 7.36.9).
o	RESPONSE_CODES	Elément lexical (voir 7.36.14).

7.35 WAVEFORM

7.35.1 Structure générale

Objet

WAVEFORM décrit un ensemble de données qui peut être affiché par un attribut GRAPH.

Structure lexicale

WAVEFORM identifier [TYPE, EMPHASIS, EXIT_ACTIONS, HANDLING, HELP, INIT_ACTIONS, KEY_POINTS, LABEL, LINE_COLOR, LINE_TYPE, REFRESH_ACTIONS, VALIDITY, VISIBILITY, Y_AXIS]

Les attributs WAVEFORM sont spécifiés dans le Tableau 182.

Tableau 182 – Attributs WAVEFORM

Utilisa-tion	Attribut	Description
m	TYPE	Elément lexical (voir 7.35.2.1).
o	EMPHASIS	Elément lexical (voir 7.36.5).
o	EXIT_ACTIONS	Elément lexical (voir 7.35.2.2).
o	HANDLING	Elément lexical (voir 7.36.6).
o	HELP	Elément lexical (voir 7.36.8).
o	INIT_ACTIONS	Elément lexical (voir 7.35.2.3).
o	KEY_POINTS	Elément lexical (voir 7.35.2.4).
o	LABEL	Elément lexical (voir 7.36.9).
o	LINE_COLOR	Elément lexical (voir 7.36.10).
o	LINE_TYPE	Elément lexical (voir 7.36.11).
o	REFRESH_ACTIONS	Elément lexical (voir 7.35.2.5).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).
o	Y_AXIS	Elément lexical (voir 7.35.2.6).

Pour chaque attribut WAVEFORM ne spécifiant pas un attribut LINE_TYPE ou LINE_COLOR, chaque ligne doit adopter une apparence différente des autres dans GRAPH.

7.35.2 Attributs spécifiques

7.35.2.1 TYPE

7.35.2.1.1 Structure générale

Objet

L'attribut TYPE spécifie le type de données contenues dans la WAVEFORM.

Structure lexicale

TYPE [XY, YT, HORIZONTAL, VERTICAL]

Les attributs TYPE sont spécifiés dans le Tableau 183.

Tableau 183 – Attributs TYPE

Utilisa-tion	Attribut	Description
s	XY	Elément lexical (voir 7.35.2.1.2.1).
s	YT	Elément lexical (voir 7.35.2.1.2.2).
s	HORIZONTAL	Elément lexical (voir 7.35.2.1.2.3).
s	VERTICAL	Elément lexical (voir 7.35.2.1.2.4).

7.35.2.1.2 Attributs spécifiques

7.35.2.1.2.1 XY

Objet

L'attribut XY spécifie un attribut WAVEFORM qui contient une liste de points (x,y).

Structure lexicale

XY [X_VALUES, Y_VALUES, NUMBER_OF_POINTS]

Les attributs XY sont spécifiés dans le Tableau 184.

Tableau 184 – Attributs XY

Utilisa-tion	Attribut	Description
m	X_VALUES	Spécifie la coordonnée X de chaque point dans l'attribut WAVEFORM. Chaque coordonnée X spécifiée dans X_VALUES doit correspondre à une coordonnée Y spécifiée dans Y_VALUES. Structure lexicale X_VALUES [reference, double, float, integer, unsigned_integer]+ Les attributs X_VALUES sont spécifiés dans le Tableau 190.
m	Y_VALUES	Spécifie la coordonnée Y de chaque point dans l'attribut WAVEFORM. Chaque coordonnée Y spécifiée dans Y_VALUES doit correspondre à une coordonnée X spécifiée dans X_VALUES. Structure lexicale Y_VALUES [(reference, double, float, integer, unsigned_integer)+ Les attributs Y_VALUES sont spécifiés dans le Tableau 190.
o	NUMBER_OF_POINTS	Spécifie le nombre de points de données valides dans X_VALUES et Y_VALUES. Par défaut, le nombre de points dans un attribut WAVEFORM sans attribut NUMBER_OF_POINTS est égal à la taille de X_VALUES et Y_VALUES Structure lexicale NUMBER_OF_POINTS (integer)<exp>

7.35.2.1.2.2 YT**Objet**

L'attribut YT spécifie un attribut WAVEFORM qui contient une liste de points qui sont définis par une coordonnée X initiale, un incrément X entre des points successifs et une liste de valeurs Y.

Structure lexicale

YT [X_INITIAL, X_INCREMENT, Y_VALUES, NUMBER_OF_POINTS]

Les attributs YT sont spécifiés dans le Tableau 185.

Tableau 185 – Attributs YT

Utilisa-tion	Attribut	Description
m	X_INITIAL	Spécifie la coordonnée X du premier point dans l'attribut WAVEFORM. Structure lexicale X_INITIAL (reference)<exp> X_INITIAL (double)<exp> X_INITIAL (float)<exp> X_INITIAL (integer)<exp> X_INITIAL (unsigned_integer)<exp> La référence est une référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY ou un élément d'une instance de REFERENCE_ARRAY.
m	X_INCREMENT	Spécifie la différence entre les coordonnées X de points adjacents dans l'attribut WAVEFORM. Structure lexicale X_INCREMENT (reference)<exp> X_INCREMENT (double)<exp> X_INCREMENT (float)<exp> X_INCREMENT (integer)<exp> X_INCREMENT (unsigned_integer)<exp> La référence est une référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY ou un élément d'une instance de REFERENCE_ARRAY.
m	Y_VALUES	Spécifie la coordonnée Y de chaque point dans l'attribut WAVEFORM. Structure lexicale Y_VALUES [(reference, double, float, integer, unsigned_integer)<exp>]+ Les attributs Y_VALUES sont spécifiés dans le Tableau 190.
o	NUMBER_OF_POINTS	Spécifie le nombre de points de données valides dans X_VALUES. Par défaut, le nombre de points dans un attribut WAVEFORM sans attribut NUMBER_OF_POINTS est égal à la taille de X_VALUES. Structure lexicale NUMBER_OF_POINTS (integer)<exp>

7.35.2.1.2.3 HORIZONTAL

Objet

L'attribut HORIZONTAL spécifie un attribut WAVEFORM qui contient des lignes horizontales.

Structure lexicale

HORIZONTAL Y_VALUES

L'attribut HORIZONTAL est spécifié dans le Tableau 186.

Tableau 186 – Attribut HORIZONTAL

Utilisa-tion	Attribut	Description
m	Y_VALUES	Spécifie la coordonnée Y de chaque ligne horizontale dans l'attribut WAVEFORM. Structure lexicale Y_VALUES [(reference, double, float, integer, unsigned_integer)+ Les attributs Y_VALUES sont spécifiés dans le Tableau 190.

7.35.2.1.2.4 VERTICAL**Objet**

L'attribut VERTICAL spécifie un attribut WAVEFORM qui contient des lignes verticales.

Structure lexicale

VERTICAL X_VALUES

L'attribut VERTICAL est spécifié dans le Tableau 187.

Tableau 187 – Attribut VERTICAL

Utilisa-tion	Attribut	Description
m	X_VALUES	Spécifie la coordonnée X de chaque ligne verticale dans l'attribut WAVEFORM. Structure lexicale X_VALUES [reference, double, float, integer, unsigned_integer)+ Les attributs X_VALUES sont spécifiés dans le Tableau 190.

7.35.2.2 EXIT_ACTIONS**Objet**

L'attribut EXIT_ACTIONS spécifie les actions qui sont exécutées lorsque l'attribut GRAPH contenant l'attribut WAVEFORM est fermé.

Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD restants ne doivent pas être exécutés.

L'attribut EXIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

EXIT_ACTIONS [reference]+, DEFINITION

Les attributs EXIT_ACTIONS sont spécifiés dans le Tableau 188.

7.35.2.3 INIT_ACTIONS

Objet

L'attribut INIT_ACTIONS spécifie les actions qui sont exécutées avant que WAVEFORM ne soit initialement affiché par un attribut GRAPH. Ces méthodes sont généralement utilisées pour charger WAVEFORM avec les données provenant d'un appareil ou d'un attribut FILE.

Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD restants ne doivent pas être exécutés. De plus, l'application EDD ne doit pas dessiner l'attribut WAVEFORM si METHOD se termine pour une raison imprévue.

L'attribut INIT_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

INIT_ACTIONS [reference]+, DEFINITION

Les attributs INIT_ACTIONS sont spécifiés dans le Tableau 188.

Tableau 188 – Attributs EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS

Utilisa-tion	Attribut	Description
o	reference	Référence à une instance de METHOD ou à une instance de METHOD avec arguments.
o	DEFINITION	Élément lexical (voir 7.36.4).

7.35.2.4 KEY_POINTS

7.35.2.4.1 Structure générale

Objet

L'attribut KEY_POINTS spécifie les points clés dans l'attribut WAVEFORM qu'il convient de mettre en évidence à l'aide de l'application EDD. Les points clés peuvent ne pas correspondre directement aux points de données spécifiés via l'attribut TYPE. La façon selon laquelle ces points sont mis en surveillance est définie par la spécification de l'EDD. Par défaut, il convient de ne mettre aucun point de WAVEFORM sans attribut KEY_POINTS en évidence.

Structure lexicale

KEY_POINTS [X_VALUES, Y_VALUES]

Les attributs KEY_POINTS sont spécifiés dans le Tableau 189.

Tableau 189 – Attributs KEY_POINTS

Utilisa-tion	Attribut	Description
m	X_VALUES	Coordonnée X de chaque KEY_POINT.
m	Y_VALUES	Coordonnée Y de chaque KEY_POINT.

7.35.2.4.2 Attributs spécifiques

7.35.2.4.2.1 X_VALUES

Objet

L'attribut X_VALUES spécifie la coordonnée X de chaque point qui doit être mise en évidence. Chaque coordonnée X spécifiée dans X_VALUES doit correspondre à une coordonnée Y spécifiée dans Y_VALUES.

Structure lexicale

X_VALUES [reference, double, float, integer, unsigned_integer] +

Les attributs X_VALUES sont spécifiés dans le Tableau 190.

Tableau 190 – Attributs X_VALUES, Y_VALUES

Utilisation	Attribut	Description
o	reference	Référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY, un élément d'une instance de REFERENCE_ARRAY, une instance de VALUE_ARRAY ou une instance de REFERENCE_ARRAY.
o	double	Constante à virgule flottante à double précision.
o	float	Constante à virgule flottante à simple précision.
o	integer	Constante entière.
o	unsigned_integer	Constante entière non signée.

7.35.2.4.2.2 Y_VALUES

Objet

L'attribut Y_VALUES spécifie la coordonnée Y de chaque point qui doit être mise en évidence. Chaque coordonnée Y spécifiée dans Y_VALUES doit correspondre à une coordonnée X spécifiée dans X_VALUES.

Structure lexicale

Y_VALUES [reference, double, float, integer, unsigned_integer] +

Les attributs Y_VALUES sont spécifiés dans le Tableau 190.

7.35.2.5 REFRESH_ACTIONS

Objet

L'attribut REFRESH_ACTIONS indique des actions qui sont exécutées lorsque des modifications relatives à X_AXIS ou Y_AXIS sont apportées ou lorsqu'un attribut CYCLE_TIME est défini dans GRAPH chaque fois que CYCLE_TIME s'écoule.

Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD restants ne doivent pas être exécutés. De plus, les attributs X_VALUES et Y_VALUES de WAVEFORM doivent être restaurés à leurs valeurs originales par l'application EDD, et l'attribut WAVEFORM sur GRAPH ne doit pas être redessiné.

L'attribut REFRESH_ACTIONS ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT. Si un comportement conditionnel est exigé, les méthodes appelées par l'action doivent être compatibles avec celui-ci.

Structure lexicale

REFRESH_ACTIONS [reference]+, DEFINITION

Les attributs REFRESH_ACTIONS sont spécifiés dans le Tableau 188.

7.35.2.6 Y_AXIS

Objet

L'attribut Y_AXIS spécifie le mode d'affichage de l'axe Y de l'attribut WAVEFORM. Chaque attribut WAVEFORM sur un attribut GRAPH partage un attribut X_AXIS commun mais dispose de son propre attribut Y_AXIS. Par défaut, l'affichage de l'axe Y d'un attribut GRAPH sans attribut Y_AXIS est dérivé des données dans l'attribut WAVEFORM.

Structure lexicale

Y_AXIS (reference)<exp>

L'attribut Y_AXIS est spécifié dans le Tableau 191.

Tableau 191 – Attribut Y_AXIS

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance d'AXIS.

7.36 Attributs communs

7.36.1 CLASSIFICATION

Objet

L'attribut CLASSIFICATION est une valeur énumérée unique qui indique les niveaux (éventuellement multiples) de classification pour cet appareil.

Structure lexicale

CLASSIFICATION [voir les attributs dans le Tableau 192]

Les attributs CLASSIFICATION sont spécifiés dans le Tableau 192.

Tableau 192 – Attributs CLASSIFICATION

Utilisa- tion	Attribut	Description
s	ACTUATOR	Le composant représente la chaîne.
s	ACTUATOR_ELECTRO_PNEUMATIC	Le composant représente la chaîne.
s	ACTUATOR_ELECTRIC	Le composant représente la chaîne.
s	ACTUATOR_HYDRAULIC	Le composant représente la chaîne.
s	CONVERTER	Le composant représente la chaîne.
s	CONTROLLER	Le composant représente la chaîne.
s	DISCRETE_IN	Le composant représente la chaîne.
s	DISCRETE_OUT	Le composant représente la chaîne.
s	ELECTRICAL_DISTRIBUTION	Le composant représente la chaîne.
s	ELECTRICAL_DISTRIBUTION_POWER_MONITORING	Le composant représente la chaîne.
s	FREQUENCY_CONVERTER	Le composant représente la chaîne.
s	INDICATOR	Le composant représente la chaîne.
s	NETWORK	Le composant représente la chaîne.
s	NETWORK_COMMUNICATION_SERVICE_PROVIDER	Le composant représente la chaîne.
s	NETWORK_COMPONENT	Le composant représente la chaîne.
s	NETWORK_COMPONENT_WIRELESSADAPTER	Le composant représente la chaîne.
s	NETWORK_CONNECTION_POINT	Le composant représente la chaîne.
s	REMOTEIO	Le composant représente la chaîne.
s	RECORDER	Le composant représente la chaîne.
s	SENSOR	Le composant représente la chaîne.
s	SENSOR_ACOUSTIC	Le composant représente la chaîne.
s	SENSOR_ANALYTIC	Le composant représente la chaîne.
s	SENSOR_ANALYTIC_GAS	Le composant représente la chaîne.
s	SENSOR_ANALYTIC_LIQUID	Le composant représente la chaîne.
s	SENSOR_CONCENTRATION	Le composant représente la chaîne.
s	SENSOR_DENSITY	Le composant représente la chaîne.
s	SENSOR_DENSITY_RADIOMETRIC	Le composant représente la chaîne.
s	SENSOR_FLOW	Le composant représente la chaîne.
s	SENSOR_FLOW_CORIOLIS	Le composant représente la chaîne.
s	SENSOR_FLOW_ELECTRO_MAGNETIC	Le composant représente la chaîne.
s	SENSOR_FLOW_MECHANICAL	Le composant représente la chaîne.
s	SENSOR_FLOW_RADIOMETRIC	Le composant représente la chaîne.
s	SENSOR_FLOW_THERMAL	Le composant représente la chaîne.
s	SENSOR_FLOW_ULTRASONIC	Le composant représente la chaîne.
s	SENSOR_FLOW_VORTEX_COUNTER	Le composant représente la chaîne.
s	SENSOR_LEVEL	Le composant représente la chaîne.
s	SENSOR_LEVEL_BUOYANCY	Le composant représente la chaîne.
s	SENSOR_LEVEL_CAPACITIVE	Le composant représente la chaîne.
s	SENSOR_LEVEL_ECHO	Le composant représente la chaîne.
s	SENSOR_LEVEL_HYDROSTATIC	Le composant représente la chaîne.
s	SENSOR_LEVEL_RADIOMETRIC	Le composant représente la chaîne.
s	SENSOR_MULTIVARIABLE	Le composant représente la chaîne.

Utilisa- tion	Attribut	Description
s	SENSOR_POSITION	Le composant représente la chaîne.
s	SENSOR_PRESSURE	Le composant représente la chaîne.
s	SENSOR_TEMPERATURE	Le composant représente la chaîne.
s	SEMICONDUCTOR	Le composant représente la chaîne.
s	SWITCHGEAR	Le composant représente la chaîne.
s	UNIVERSAL	Le composant représente la chaîne.
s	string	Le composant représente la chaîne.

7.36.2 COMPONENT_PARENT

Objet

L'attribut COMPONENT_PARENT est une référence à un attribut COMPONENT_FOLDER. COMPONENT_PARENT peut être utilisé à la place de PROTOCOL et CLASSIFICATION pour simplifier (raccourcir) la déclaration de COMPONENT_PATH. Si l'attribut PROTOCOL et/ou l'attribut CLASSIFICATION sont définis dans COMPONENT_PARENT (ou son parent), PROTOCOL et/ou CLASSIFICATION ne doivent pas être redéfinis. Si l'attribut COMPONENT_PARENT est spécifié, l'attribut COMPONENT_PATH doit être défini.

NOTE COMPONENT_PARENT est prédéfini par la structure de bibliothèque d'EDD HCF et FF.

Structure lexicale

COMPONENT_PARENT reference

L'attribut COMPONENT_PARENT est spécifié dans le Tableau 193.

Tableau 193 – Attribut COMPONENT_PARENT

Utilisa- tion	Attribut	Description
m	reference	Composant parent dans le catalogue.

7.36.3 COMPONENT_PATH

Objet

L'attribut COMPONENT_PATH permet au fabricant d'EDD de désigner l'emplacement dans le catalogue d'une application EDD. Le chemin spécifie uniquement la partie spécifique au fabricant de l'emplacement sans l'attribut PROTOCOL, CLASSIFICATION et MANUFACTURER. Si COMPONENT_PARENT est utilisé, les attributs PROTOCOL, CLASSIFICATION et MANUFACTURER peuvent ne pas être définis et COMPONENT_PATH définit uniquement la sous-hiérarchie.

NOTE COMPONENT_PATH est prédéfini par la structure de bibliothèque d'EDD HCF et FF.

Structure lexicale

COMPONENT_PATH string

L'attribut COMPONENT_PATH est spécifié dans le Tableau 194.

Tableau 194 – Attribut COMPONENT_PATH

Utilisa-tion	Attribut	Description
m	string	Chemin d'accès spécifique au fabricant du composant dans le catalogue.

7.36.4 DEFINITION**Objet**

L'attribut DEFINITION spécifie les actions à effectuer et est basé sur un langage de programmation procédural (par exemple, langage de programmation C). Dans les attributs DEFINITION, des attributs METHOD et des Builtins (voir IEC 61804-5) peuvent être appelés. Des éléments de base de l'EDD peuvent être utilisés dans un attribut DEFINITION.

Les éléments suivants doivent être pris en charge.

- Types de données de base (voir D.6.1).
- Matrices.
- Opérateurs (voir A.3).
- Instructions.
- Invocation d'attributs METHOD et de Builtins.
- METHOD avec arguments (appel par valeur, appel par référence):
 - Un appel par valeur est utilisé pour transmettre une copie des données à l'attribut METHOD. Des changements de la copie dans METHOD ne modifient pas les données originales.
 - L'appel par référence est utilisé pour accéder directement aux données. Les changements affectent les données originales.

Structure lexicale

DEFINITION compound-statement

L'attribut DEFINITION est spécifié dans le Tableau 195.

Tableau 195 – Attribut DEFINITION

Utilisa-tion	Attribut	Description
o	compound-statement	Bloc d'instructions pour un langage de programmation procédural pris en charge.

7.36.5 EMPHASIS**Objet**

L'attribut EMPHASIS spécifie s'il convient de mettre un élément d'interface utilisateur en évidence lorsqu'il est affiché. La façon selon laquelle il est mis en évidence est définie par l'application EDD. Par défaut, il convient de ne pas mettre en évidence un élément d'interface utilisateur sans attribut EMPHASIS.

Structure lexicale

EMPHASIS (FALSE, TRUE) <exp>

Les attributs EMPHASIS sont spécifiés dans le Tableau 196.

Tableau 196 – Attributs EMPHASIS

Utilisa-tion	Attribut	Description
s	FALSE	Il convient de ne pas souligner l'élément d'interface utilisateur.
s	TRUE	Il convient de souligner l'élément d'interface utilisateur.

7.36.6 HANDLING**Objet**

L'attribut HANDLING spécifie les opérations qui peuvent être effectuées sur un élément. Par défaut, un élément sans attribut HANDLING peut être lu et écrit.

Structure lexicale

HANDLING ([READ, WRITE]+) <exp>

Les attributs HANDLING sont spécifiés dans le Tableau 197.

Tableau 197 – Attributs HANDLING

Utilisa-tion	Attribut	Description
s	READ	L'élément peut être lu.
s	WRITE	L'élément peut être écrit.

7.36.7 HEIGHT**Objet**

L'attribut HEIGHT spécifie la taille verticale d'un élément d'interface utilisateur lorsqu'il est affiché. Un attribut VARIABLE sans attribut HEIGHT défini est traité comme s'il disposait de l'attribut HEIGHT XXX_SMALL. Les attributs GRAPH, CHART et GRID sans attribut HEIGHT spécifié sont traités comme s'ils avaient l'attribut HEIGHT MEDIUM.

Structure lexicale

HEIGHT (XXX_SMALL, XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE)

Les attributs HEIGHT sont spécifiés dans le Tableau 198.

Tableau 198 – Attributs HEIGHT/WIDTH

Utilisa-tion	Attribut	Description
s	XXX_SMALL	Il convient que l'élément d'interface utilisateur occupe une partie extrêmement petite de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe une colonne. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe la hauteur minimale possible pour un champ d'entrée normalisé. XXX_SMALL n'est pas admis pour les attributs VARIABLE.
s	XX_SMALL	Il convient que l'élément d'interface utilisateur occupe une partie extrêmement petite de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe une colonne. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe trois fois la hauteur minimale possible pour un champ d'entrée normalisé.
s	X_SMALL	Il convient que l'élément d'interface utilisateur occupe une très petite partie de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe une colonne. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe cinq fois la hauteur minimale possible pour un champ d'entrée normalisé.
s	SMALL	Il convient que l'élément d'interface utilisateur occupe une petite partie de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe deux colonnes. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe sept fois la hauteur minimale possible pour un champ d'entrée normalisé.
s	MEDIUM	Il convient que l'élément d'interface utilisateur occupe une partie moyenne de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe trois colonnes. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe huit fois la hauteur minimale possible pour un champ d'entrée normalisé.
s	LARGE	Il convient que l'élément d'interface utilisateur occupe une grande partie de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe quatre colonnes. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe neuf fois la hauteur minimale possible pour un champ d'entrée normalisé.
s	X_LARGE	Il convient que l'élément d'interface utilisateur occupe une très grande partie de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe cinq colonnes. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe onze fois la hauteur minimale possible pour un champ d'entrée normalisé.
s	XX_LARGE	Il convient que l'élément d'interface utilisateur occupe une partie extrêmement grande de la fenêtre dans laquelle il est affiché. Si cette valeur est utilisée pour WIDTH, l'élément d'interface occupe cinq colonnes. Si cette valeur est utilisée pour HEIGHT, l'élément d'interface occupe treize fois la hauteur minimale possible pour un champ d'entrée normalisé.

7.36.8 HELP**Objet**

L'attribut HELP spécifie du texte, qui donne une description de la construction.

NOTE Ce texte est destiné à être utilisé par des applications EDD pour information des utilisateurs.

Structure lexicale

HELP (string) <exp>

L'attribut HELP est spécifié dans le Tableau 199.

Tableau 199 – Attribut HELP

Utilisa-tion	Attribut	Description
m	string	Texte lisible par l'utilisateur.

7.36.9 LABEL**Objet**

L'attribut LABEL spécifie la désignation affichée d'un élément.

Structure lexicale

`LABEL (string)<exp>`

L'attribut LABEL est spécifié dans le Tableau 200.

Tableau 200 – Attribut LABEL

Utilisa-tion	Attribut	Description
m	string	Texte lisible par l'utilisateur.

7.36.10 LINE_COLOR**Objet**

L'attribut LINE_COLOR spécifie de manière facultative la couleur qu'il convient d'utiliser pour afficher un élément d'interface utilisateur. Un élément d'interface utilisateur sans attribut LINE_COLOR sera affiché en utilisant une couleur par défaut définie par l'application EDD.

Structure lexicale

`LINE_COLOR (integer, reference)<exp>`

Les couleurs sont spécifiées par l'intermédiaire d'une valeur RGB, comme défini dans W3C Cascading Style Sheets, niveau 2.

Les attributs LINE_COLOR sont spécifiés dans le Tableau 201.

Tableau 201 – Attributs LINE_COLOR

Utilisa-tion	Attribut	Description
s	integer	Couleur à utiliser pour afficher l'élément d'interface utilisateur.
s	reference	Référence à une instance de VARIABLE contenant la couleur à utiliser pour afficher l'élément d'interface utilisateur.

7.36.11 LINE_TYPE**Objet**

L'attribut LINE_TYPE est utilisé pour différencier les types de données par couleur, motif ou épaisseur de ligne, etc. Dans un élément d'interface utilisateur, il convient d'afficher les éléments contenus dans le même attribut LINE_TYPE, sauf pour DATA, de la même manière.

Par défaut, un élément sans attribut LINE_TYPE utilise le type de ligne DATA. Pour le type de ligne DATA (pas de numéro), l'application EDD doit afficher les éléments de manière différente.

Structure lexicale

```
LINE_TYPE ([DATA, n]*, LOW_LOW_LIMIT, LOW_LIMIT, HIGH_LIMIT, HIGH_HIGH_LIMIT, TRANSPARENT) <exp>
```

Les attributs LINE_TYPE sont spécifiés dans le Tableau 202.

Tableau 202 – Attribut LINE_TYPE

Utilisa-tion	Attribut	Description
s	DATA	L'élément d'interface utilisateur contient des données.
c	n	Entier entre 1 et 9, qui identifie des types de ligne DATA distincts si plusieurs sont présents.
s	LOW_LOW_LIMIT	L'élément d'interface utilisateur contient une limite basse inférieure.
s	LOW_LIMIT	L'élément d'interface utilisateur contient une limite basse.
s	HIGH_LIMIT	L'élément d'interface utilisateur contient une limite haute.
s	HIGH_HIGH_LIMIT	L'élément d'interface utilisateur contient une limite haute supérieure.
s	TRANSPARENT	Seuls les points de données d'élément d'interface utilisateur sont visibles.

7.36.12 MEMBERS

Objet

L'attribut MEMBERS définit les membres d'une construction de base. Chaque membre spécifie un élément dans le groupe et est défini par un ensemble de quatre attributs (identificateur, référence, description, aide).

L'attribut MEMBERS de FILE ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT.

Structure lexicale

```
MEMBERS [(identifiant, reference, description, help) <exp>]+
```

Les attributs MEMBERS sont spécifiés dans le Tableau 203.

Tableau 203 – Attributs MEMBERS

Utilisa-tion	Attribut	Description
m	identifier	Nom par lequel l'élément peut être référencé. Un nom d'identificateur spécifique peut être réutilisé dans l'attribut MEMBERS d'éléments multiples dans la mesure où les éléments sont du même type.
m	reference	Référence à un élément EDD. Le type de la référence dépend des constructions de base: ▪ CHART: référence à une instance de SOURCE; ▪ COLLECTION: référence à une instance de BLOCK_A, BLOCK_B, CHART, COLLECTION, EDIT_DISPLAY, FILE, GRAPH, LIST, MENU, METHOD, PLUGIN, RECORD, REFERENCE_ARRAY, VALUE_ARRAY, VARIABLE ou VARIABLE_LIST; ▪ FILE ou LIST: référence à une instance de COLLECTION, LIST, RECORD, REFERENCE_ARRAY, VALUE_ARRAY ou VARIABLE. Pour COLLECTION, le type de COLLECTION doit être LIST, RECORD, REFERENCE_ARRAY, VALUE_ARRAY, ou VARIABLE; ▪ GRAPH: référence à une instance de WAVEFORM; ▪ RECORD: référence à une instance de VARIABLE; ▪ SOURCE: référence à une instance de VARIABLE; ▪ VARIABLE_LIST: référence à une instance de RECORD, VALUE_ARRAY ou VARIABLE.
o	description	Brève description de l'élément.
o	help	Texte d'aide pour l'élément.

7.36.13 PROTOCOL

Objet

L'attribut PROTOCOL indique le protocole de communication utilisé pour échanger avec l'appareil modélisé par l'EDD en question. L'attribut PROTOCOL est utilisé pour séparer les EDD dans le catalogue.

Structure lexicale

PROTOCOL [PROFIBUS_DP, PROFIBUS_PA, PROFINET, FF, HART, string]

Les attributs PROTOCOL sont spécifiés dans le Tableau 204.

Tableau 204 – Attributs PROTOCOL

Utilisa-tion	Attribut	Description
s	FF	Le composant est un composant Fieldbus Foundation.
s	HART	Le composant est un composant HART.
s	PROFIBUS_DP	Le composant est un composant PROFIBUS DP.
s	PROFIBUS_PA	Le composant est un composant PROFIBUS PA.
s	PROFINET	Le composant est un composant PROFINET.
s	string	Spécifie un protocole quelconque. Le protocole doit être pris en charge par l'application EDD utilisée.

7.36.14 RESPONSE_CODES

Objet

L'attribut RESPONSE_CODES spécifie les valeurs qu'un appareil peut retourner en tant qu'information d'erreur. Chaque attribut VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST ou COMMAND peut avoir son propre ensemble de RESPONSE_CODES.

Deux structures lexicales pour l'attribut RESPONSE_CODES sont fournies:

- l'attribut RESPONSE_CODES référencé contient une référence à une instance de RESPONSE_CODES;
- l'attribut RESPONSE_CODES imbriqué permet de spécifier les éléments RESPONSE_CODES directement à l'instance d'élément.

Structure lexicale pour les attributs RESPONSE_CODES référencés

RESPONSE_CODES (reference) <exp>

L'attribut RESPONSE_CODES référencé est spécifié dans le Tableau 205.

Tableau 205 – Attribut RESPONSE_CODES

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de RESPONSE_CODES.

Structure lexicale pour les attributs RESPONSE_CODES imbriqués

RESPONSE_CODES [(description, integer, [DATA_ENTRY_ERROR, DATA_ENTRY_WARNING, MISC_ERROR, MISC_WARNING, MODE_ERROR, PROCESS_ERROR, SUCCESS], help) <exp>]+

Les attributs RESPONSE_CODES imbriqués sont spécifiés dans le Tableau 157.

7.36.15 SUPPLIED_INTERFACE

Objet

L'attribut SUPPLIED_INTERFACE répertorie toutes les interfaces prises en charge par ce composant.

Structure lexicale

SUPPLIED_INTERFACE [interface-reference]+

L'attribut SUPPLIED_INTERFACE est spécifié dans le Tableau 206.

Tableau 206 – Attribut SUPPLIED_INTERFACE

Utilisa-tion	Attribut	Description
m	interface-reference	Liste d'identificateurs d'interface.

7.36.16 VALIDITY

Objet

L'attribut VALIDITY spécifie si un élément est ou n'est pas valide. Un élément sans attribut VALIDITY est toujours valide.

Il convient que VALIDITY soit exprimé à l'aide d'une instruction conditionnelle IF, IF-ELSE, SELECT. Si un élément d'interface utilisateur n'est pas valide, il convient de ne pas l'afficher. Une METHOD doit être interrompue lorsqu'elle accède à un attribut VARIABLE non valide.

Structure lexicale

VALIDITY [(FALSE, TRUE)<exp>]

Les attributs VALIDITY sont spécifiés dans le Tableau 207.

Tableau 207 – Attributs VALIDITY

Utilisation	Attribut	Description
s	FALSE	L'élément n'est pas valide.
s	TRUE	L'élément est valide.

7.36.17 WIDTH

Objet

L'attribut WIDTH spécifie la taille horizontale d'un élément d'interface utilisateur lorsqu'il est affiché. Un attribut VARIABLE sans attribut WIDTH spécifié est traité comme s'il disposait de l'attribut WIDTH XXX_SMALL. Les attributs GRAPH, CHART et GRID sans attribut WIDTH spécifié sont traités comme s'ils avaient un attribut WIDTH MEDIUM.

Structure lexicale

WIDTH (XXX_SMALL, XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE)

Les attributs WIDTH sont spécifiés dans le Tableau 198.

7.36.18 PRIVATE

Objet

L'attribut PRIVATE spécifie si un élément est exposé en dehors de l'application EDD. Si l'attribut PRIVATE n'est pas spécifié, l'élément est toujours public.

Structure lexicale

PRIVATE [(FALSE, TRUE)]

Les attributs PRIVATE sont spécifiés dans le Tableau 208.

Tableau 208 – Attributs PRIVATE

Utilisation	Attribut	Description
s	FALSE	L'élément est public. La valeur par défaut de PRIVATE est FALSE.
s	TRUE	L'élément est privé.

7.36.19 VISIBILITY

Objet

L'attribut VISIBILITY spécifie si un élément est ou n'est pas visible. Un élément sans VALIDITY est toujours valide.

Il convient que VISIBILITY soit exprimé à l'aide d'une instruction conditionnelle IF, IF-ELSE, SELECT. VISIBILITY n'affecte pas la disposition de l'écran. Dans la disposition de l'écran, le même espace est réservé, comme si l'élément était affiché.

Structure lexicale

VISIBILITY [(FALSE, TRUE)<exp>]

Les attributs VISIBILITY sont spécifiés dans le Tableau 209.

Tableau 209 – Attributs VISIBILITY

Utilisation	Attribut	Description
s	FALSE	L'élément n'est pas visible.
s	TRUE	L'élément est visible. La valeur par défaut de VISIBILITY est TRUE.

7.36.20 WRITE_MODE

Objet

L'attribut WRITE_MODE spécifie les exigences minimales du mode cible qui doivent être définies pour que la valeur puisse être transférée à l'appareil.

Structure lexicale

WRITE_MODE [(MODE_OUT_OF_SERVICE, MODE_INITIALIZATION, MODE_LOCAL_OVERRIDE, MODE_MANUAL, MODE_AUTOMATIC, MODE_CASCADE, MODE_REMOTE_CASCADE, MODE_REMOTE_OUTPUT)<exp>]

Les attributs WRITE_MODE sont spécifiés dans le Tableau 210.

Tableau 210 – Attributs WRITE_MODE

Utilisation	Attribut	Priorité	Description
s	MODE_OUT_OF_SERVICE	7 – la plus élevée	Out of Service (O/S).
s	MODE_INITIALIZATION	6	Initialization Manual (IMan)
s	MODE_LOCAL_OVERRIDE	5	Local Override (LO)
s	MODE_MANUAL	4	Manual (Man)
s	MODE_AUTOMATIC	3	Automatic (Auto)
s	MODE_CASCADE	2	Cascade (Cas)
s	MODE_REMOTE_CASCADE	1	Remote-Cascade (RCas)
s	MODE_REMOTE_OUTPUT	0 – la moins élevée	Remote-Output (ROut)

7.36.21 IDENTITY

Objet

L'attribut IDENTITY spécifie un nom de fichier. La spécification d'IDENTITY permet d'accéder à des fichiers qui peuvent être créés ou utilisés par des applications externes à l'application EDD.

Structure lexicale

IDENTITY file-name

L'attribut IDENTITY est spécifié dans le Tableau 211.

Tableau 211 – Attribut IDENTITY

Utilisa-tion	Attribut	Description
m	file-name	Littéral de chaîne qui identifie le nom de fichier.

7.37 Expression conditionnelle

Objet

Des expressions conditionnelles sont utilisées pour déclarer des valeurs alternatives ou calculer des valeurs pour un attribut d'un élément de base. La valeur de l'expression est évaluée pendant le temps d'exécution. L'instruction conditionnelle IF ou SELECT peut être utilisée pour activer ou désactiver les instances référencées d'une liste d'attributs ITEMS de MENU, etc.

Trois types d'expressions conditionnelles peuvent être utilisés.

- Une instruction conditionnelle IF est utilisée pour spécifier un attribut qui a deux définitions alternatives. L'expression spécifiée dans l'instruction conditionnelle IF est évaluée. Si le résultat est différent de zéro, l'attribut est spécifié par un article then-clause ("alors"); sinon, il est spécifié par else-clause ("sinon").
- Une instruction conditionnelle SELECT est utilisée pour spécifier un attribut qui a plusieurs définitions alternatives. L'expression select-expression est évaluée et comparée à chaque integer_n de CASE. Si une correspondance est trouvée, la select-clause appropriée est valide. Si aucun integer_n correspondant n'est trouvé, l'article par défaut default-clause suivant DEFAULT est utilisé.
- Une expression primaire, unaire ou binaire (voir 7.39.7). Ce type d'expression conditionnelle peut être utilisé uniquement pour des valeurs numériques (par exemple, DEFAULT_VALUE, MIN_VALUE, SLOT, INDEX).

EXEMPLE Plusieurs valeurs peuvent être définies pour les attributs DEFAULT_VALUE, MIN_VALUE, entre autres, en utilisant une instruction conditionnelle IF ou SELECT.

Structure lexicale pour l'instruction conditionnelle IF

IF (if-expression), (then-clause)+ ELSE (else-clause)+

Les attributs sont spécifiés dans le Tableau 212.

Structure lexicale pour l'instruction conditionnelle SELECT

SELECT select-expression, (CASE integer_n, select-clause)+, (DEFAULT, default-clause)

Les attributs sont spécifiés dans le Tableau 212.

Tableau 212 – Instruction conditionnelle IF, SELECT

Utilisa-tion	Attribut	Description
m	if-expression	Évaluée pour déterminer si l'article then-clause ou else-clause est utilisé pour définir un attribut. Si l'expression if-expression contient une référence à une instance de VARIABLE, la valeur de l'attribut VARIABLE référencé est utilisée.
m	then-clause	Définition pour l'attribut si la valeur de condition est différente de zéro. La structure d'article dépend de l'attribut défini. Elle peut également prendre la forme d'une autre instruction conditionnelle.
o	else-clause	Définition pour l'attribut si la valeur de condition est zéro. La structure d'article dépend de l'attribut défini. Elle peut également prendre la forme d'une autre instruction conditionnelle.
m	select-expression	Évalué pour déterminer quel article select-clause est utilisé pour définir un attribut. Si l'expression select-expression contient une référence à une instance de VARIABLE, la valeur de l'attribut VARIABLE référencé est utilisée.
m	integer_n	Numéro qui identifie l'attribut CASE correspondant. Si un entier correspond à integer_n, la select_clause correspondante est utilisée.
m	select-clause	Définition pour l'attribut pour chaque attribut CASE, et définition de DEFAULT. La structure de chaque article dépend de l'attribut défini. Indépendamment de l'attribut, chaque article peut également prendre la forme d'une autre instruction conditionnelle.
o	default-clause	Définition pour la définition de DEFAULT. La structure de default-clause dépend de l'attribut défini. Indépendamment de l'attribut, chaque article peut également prendre la forme d'une autre instruction conditionnelle.

Si l'article

- then-clause,
- else-clause,
- select-clause,
- default-clause

contient des références à des instances de METHOD ou des Builtins, la valeur retournée de la METHOD ou du Builtin référencés doit être utilisée dans l'article. Les valeurs de retour doivent être compatibles au niveau du type.

7.38 Référencement

7.38.1 Référencement d'une instance d'EDD

Objet

Des références sont utilisées dans une EDD pour faire référence à d'autres éléments de l'EDD.

EXEMPLE Un attribut PRE_EDIT_ACTIONS de VARIABLE fait référence à des instances de METHOD définies ailleurs dans l'EDD.

Structure lexicale

L'attribut est spécifié dans le Tableau 213.

Tableau 213 – Référencement d'une instance d'EDD

Utilisa-tion	Attribut	Description
m	identifier	Identificateur d'une instance d'EDD.

7.38.2 Référencement de bits de BIT_ENUMERATED VARIABLE

Objet

Ce référencement est utilisé pour accéder à un bit d'une instance BIT_ENUMERATED VARIABLE.

Structure lexicale

variable-reference, bit-mask

Les attributs sont spécifiés dans le Tableau 214.

Tableau 214 – Référencement d'éléments de VARIABLE

Utilisa-tion	Attribut	Description
m	variable-reference	Référence à une instance de VARIABLE. Le type d'instance de VARIABLE doit être BIT_ENUMERATED.
m	bit-mask	Identifie un seul bit dans la variable. Le bit-mask doit être une expression constante.

7.38.3 Référencement des membres d'un RECORD

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de RECORD.

Structure lexicale

record-reference, member-identifier

Les attributs sont spécifiés dans le Tableau 215.

Tableau 215 – Référencement d'éléments de RECORD

Utilisa-tion	Attribut	Description
m	record-reference	Référence à une instance de RECORD.
m	member-identifier	Identificateur d'un élément de MEMBERS.

7.38.4 Référencement d'éléments d'un attribut VALUE_ARRAY

Objet

Ce référencement est utilisé pour accéder à un élément d'une instance de VALUE_ARRAY.

Structure lexicale

value-array-reference, expression

Les attributs sont spécifiés dans le Tableau 183.

Tableau 216 – Référencement d'éléments de VALUE_ARRAY

Utilisa-tion	Attribut	Description
m	value-array-reference	Référence à une instance de VALUE_ARRAY.
m	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut VALUE_ARRAY (pour une description des expressions, voir 7.40).

7.38.5 Référencement des membres d'une COLLECTION

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de COLLECTION.

Structure lexicale

collection-reference, member-identifiant

Les attributs sont spécifiés dans le Tableau 217.

Tableau 217 – Référencement des membres de COLLECTION

Utilisa-tion	Attribut	Description
m	collection-reference	Référence à une instance de COLLECTION.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.6 Référencement d'éléments d'un attribut REFERENCE_ARRAY

Objet

Ce référencement est utilisé pour accéder à un élément d'une instance de REFERENCE_ARRAY.

Structure lexicale

reference-array-reference, expression

Les attributs sont spécifiés dans le Tableau 218.

Tableau 218 – Référencement des membres de REFERENCE_ARRAY

Utilisa-tion	Attribut	Description
m	reference-array-reference	Référence à une instance de REFERENCE_ARRAY.
m	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut REFERENCE_ARRAY (pour une description des expressions, voir 7.40).

7.38.7 Référencement des membres d'un attribut VARIABLE_LISTS

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de VARIABLE_LISTS.

Structure lexicale

variable-list-reference, member-identifiant

Les attributs sont spécifiés dans le Tableau 219.

Tableau 219 – Référencement des membres de VARIABLE_LISTS

Utilisa-tion	Attribut	Description
m	variable-list-reference	Référence à une instance de VARIABLE_LISTS.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.8 Référencement d'éléments BLOCK_A PARAMETERS

Objet

Ce référencement est utilisé pour accéder à un élément de l'attribut PARAMETERS d'une instance de BLOCK_A.

Structure lexicale

PARAM, member-identifiant

L'attribut est spécifié dans le Tableau 220.

Tableau 220 – Référencement des membres de BLOCK_A PARAMETERS

Utilisa-tion	Attribut	Description
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.9 Référencement d'éléments de BLOCK_A PARAMETER_LISTS

Objet

Ce référencement est utilisé pour accéder à un élément de l'attribut PARAMETER_LISTS d'une instance de BLOCK_A.

Structure lexicale

PARAM_LIST, member-identifiant

L'attribut est spécifié dans le Tableau 221.

Tableau 221 – Référencement des membres de BLOCK_A PARAMETER_LISTS

Utilisa-tion	Attribut	Description
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.10 Référencement d'éléments de BLOCK_A LOCAL_PARAMETERS

Objet

Ce référencement est utilisé pour accéder à un élément de l'attribut LOCAL_PARAMETERS d'une instance de BLOCK_A.

Structure lexicale

LOCAL_PARAM, member-identifiant

L'attribut est spécifié dans le Tableau 222.

Tableau 222 – Référencement des membres de BLOCK_A LOCAL_PARAMETERS

Utilisa-tion	Attribut	Description
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.11 Référencement de BLOCK_A CHARACTERISTICS

Objet

Ce référencement est utilisé pour accéder aux attributs CHARACTERISTICS d'une instance de BLOCK_A.

Structure lexicale

BLOCK, characteristics-identifiant

L'attribut est spécifié dans le Tableau 223.

Tableau 223 – Référencement de BLOCK_A CHARACTERISTICS

Utilisa-tion	Attribut	Description
m	characteristics-identifiant	Identificateur de la référence de CHARACTERISTICS.

7.38.12 Référencement des membres d'un attribut FILE

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de FILE.

Structure lexicale

file-reference, member-identifiant

Les attributs sont spécifiés dans le Tableau 224.

Tableau 224 – Référencement des membres de FILE

Utilisa-tion	Attribut	Description
m	file-reference	Référence à une instance de FILE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.13 Référencement d'éléments de LIST

Objet

Ce référencement est utilisé pour accéder à un élément d'une instance de LIST.

Structure lexicale

`list-reference, (expression, FIRST, LAST)`

Les attributs sont spécifiés dans le Tableau 225.

Tableau 225 – Référencement d'éléments de LIST

Utilisa-tion	Attribut	Description
m	list-reference	Référence à une instance de LIST.
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut LIST (pour une description des expressions, voir 7.39.7).
s	FIRST	Premier élément de LIST.
s	LAST	Dernier élément de LIST.

7.38.14 Référencement des membres de CHART

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de CHART.

Structure lexicale

`chart-reference, member-identifiser`

Les attributs sont spécifiés dans le Tableau 226.

Tableau 226 – Référencement des membres de CHART

Utilisa-tion	Attribut	Description
m	chart-reference	Référence à une instance de CHART.
m	member-identifiser	Identificateur d'un élément de MEMBERS.

7.38.15 Référencement de membres d'un attribut GRAPH

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de GRAPH.

Structure lexicale

`graph-reference, member-identifiser`

Les attributs sont spécifiés dans le Tableau 227.

Tableau 227 – Référencement des membres de GRAPH

Utilisa-tion	Attribut	Description
m	graph-reference	Référence à une instance de GRAPH.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.16 Référencement des membres de SOURCE**Objet**

Ce référencement est utilisé pour accéder à un membre d'une instance de GRAPH.

Structure lexicale

source-reference, member-identifiant

Les attributs sont spécifiés dans le Tableau 228.

Tableau 228 – Référencement des membres de SOURCE

Utilisa-tion	Attribut	Description
m	source-reference	Référence à une instance de SOURCE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.17 Référencement d'AXIS de GRAPH, SOURCE, WAVEFORM**Objet**

Ce référencement est utilisé pour accéder à un attribut AXIS de GRAPH, SOURCE ou WAVEFORM.

Structure lexicale

reference, (X_AXIS, Y_AXIS)

Les attributs sont spécifiés dans le Tableau 229.

Tableau 229 – Référencement d'AXIS d'un attribut GRAPH, SOURCE, WAVEFORM

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de GRAPH, une référence à une instance de SOURCE, ou une référence à une instance de WAVEFORM.
c/o	X_AXIS	Ne doit être utilisé qu'avec GRAPH; Y_AXIS utilisé dans tous les autres cas.
c/o	Y_AXIS	Ne doit être utilisé qu'avec SOURCE et WAVEFORM; X_AXIS utilisé dans tous les autres cas.

7.38.18 Référencement de PARAMETERS d'une instance spécifique de BLOCK_A**Objet**

Ce référencement est utilisé pour accéder à un membre de l'attribut PARAMETERS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, PARAM, member-identifier

Les attributs sont spécifiés dans le Tableau 230.

Tableau 230 – Référencement de PARAMETERS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifier	Identificateur d'un élément de MEMBERS.

7.38.19 Référencement de LOCAL_PARAMETERS d'une instance spécifique de BLOCK_A

Objet

Ce référencement est utilisé pour accéder à un membre de l'attribut LOCAL_PARAMETERS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, LOCAL_PARAM, member-identifier

Les attributs sont spécifiés dans le Tableau 231.

Tableau 231 – Référencement de LOCAL_PARAMETERS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifier	Identificateur d'un élément de MEMBERS.

7.38.20 Référencement de CHARACTERISTICS d'une instance spécifique de BLOCK_A

Objet

Ce référencement est utilisé pour accéder à un membre de l'attribut CHARACTERISTICS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, BLOCK, member-identifier

Les attributs sont spécifiés dans le Tableau 232.

**Tableau 232 – Référencement de CHARACTERISTICS
d'une instance spécifique de BLOCK_A**

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.21 Référencement de CHARTS d'une instance spécifique de BLOCK_A

Objet

Ce référencement est utilisé pour accéder à un membre de l'attribut CHARTS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, CHART, member-identifiant

Les attributs sont spécifiés dans le Tableau 233.

Tableau 233 – Référencement de CHARTS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.22 Référencement de LISTS d'une instance spécifique de BLOCK_A

Objet

Ce référencement est utilisé pour accéder à un membre de l'attribut LISTS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, LIST, member-identifiant

Les attributs sont spécifiés dans le Tableau 234.

Tableau 234 – Référencement de LISTS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.23 Référencement de GRAPHS d'une instance spécifique de BLOCK_A**Objet**

Ce référencement est utilisé pour accéder à un membre de l'attribut GRAPHS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, GRAPH, member-identifiant

Les attributs sont spécifiés dans le Tableau 235.

Tableau 235 – Référencement de GRAPHS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.24 Référencement de GRIDS d'une instance spécifique de BLOCK_A**Objet**

Ce référencement est utilisé pour accéder à un membre de l'attribut GRIDS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, GRID, member-identifiant

Les attributs sont spécifiés dans le Tableau 236.

Tableau 236 – Référencement de GRIDS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifier	Identificateur d'un élément de MEMBERS.

7.38.25 Référencement de MENUS d'une instance spécifique de BLOCK_A**Objet**

Ce référencement est utilisé pour accéder à un membre de l'attribut MENUS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, MENU, member-identifier

Les attributs sont spécifiés dans le Tableau 237.

Tableau 237 – Référencement de MENUS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifier	Identificateur d'un élément de MEMBERS.

7.38.26 Référencement d'attributs METHODS d'une instance spécifique de BLOCK_A**Objet**

Ce référencement est utilisé pour accéder à un membre de l'attribut METHODS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, METHOD, member-identifier

Les attributs sont spécifiés dans le Tableau 238.

**Tableau 238 – Référencement d'attributs METHODS
d'une instance spécifique de BLOCK_A**

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.27 Référencement des instances de COMPONENT

Objet

Ce référencement est utilisé pour référencer une instance qui est liée à un attribut COMPONENT_RELATION d'un composant. Si la référence est utilisée dans une déclaration de MENU et si l'instance n'existe pas, l'élément référencé sera traité comme étant non valide. Si la référence est utilisée dans un attribut DECLARATION de METHOD et si l'instance n'existe pas, l'attribut METHOD sera interrompu. Une telle référence ne peut pas être utilisée dans une expression conditionnelle.

Structure lexicale

reference, expression, identifier

Les attributs de référencement de COMPONENT sont spécifiés dans le Tableau 239.

Tableau 239 – Référencement d'une instance de COMPONENT

Utilisa-tion	Attribut	Description
o	reference	Référence à un attribut COMPONENT.
o	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut COMPONENT_RELATION.
o	identifier	Identificateur d'une construction de base de l'attribut COMPONENT référencé.

7.38.28 Référencement de types de COMPONENT

Objet

Ce référencement est utilisé pour référencer un type de composant. Toutes les expressions conditionnelles sont évaluées avec des attributs INITIAL_VALUE ou, si ceux-ci ne sont pas définis, DEFAULT_VALUE ou avec des valeurs par défaut de type de données.

Structure lexicale

reference, identifier

Les attributs de référencement de COMPONENT sont spécifiés dans le Tableau 240.

Tableau 240 – Référencement d'un type COMPONENT

Utilisa-tion	Attribut	Description
o	reference	Référence à un attribut COMPONENT.
o	identifiant	Identificateur d'une construction de base du COMPONENT référencé.

7.38.29 Référencement de FILES d'une instance spécifique de BLOCK_A**Objet**

Ce référencement est utilisé pour accéder à un membre de l'attribut FILES d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, FILE, member-identifiant

Les attributs sont spécifiés dans le Tableau 241.

Tableau 241 – Référencement de FILES d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.38.30 Référencement de PLUGINS d'une instance spécifique de BLOCK_A**Objet**

Ce référencement est utilisé pour accéder à un membre de l'attribut PLUGINS d'une instance spécifique de BLOCK_A.

Structure lexicale

reference, expression, PLUGIN, member-identifiant

Les attributs sont spécifiés dans le Tableau 242.

Tableau 242 – Référencement de PLUGINS d'une instance spécifique de BLOCK_A

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de BLOCK_A.
m	expression	Entier non négatif qui spécifie l'occurrence de l'instance de BLOCK_A dans l'appareil. Si le bloc spécifié n'existe pas, l'application EDD doit se comporter comme si l'attribut VALIDITY de l'élément référencé était FALSE.
m	member-identifiant	Identificateur d'un élément de MEMBERS.

7.39 Chaînes

7.39.1 Spécification d'une chaîne sous forme d'un littéral de chaîne

Objet

Une chaîne de texte peut être spécifiée sous la forme d'un littéral de chaîne (voir A.2.3).

Structure lexicale

`string`

L'attribut est spécifié dans le Tableau 243.

Tableau 243 – Chaîne sous forme de littéral de chaîne

Utilisa-tion	Attribut	Description
m	string	Littéral de chaîne.

7.39.2 Spécification d'une chaîne en tant que variable de chaîne

Objet

Une chaîne de texte spécifiée en tant que variable de chaîne possède la valeur de la variable de chaîne.

Structure lexicale

`reference`

L'attribut est spécifié dans le Tableau 244.

Tableau 244 – Chaîne en tant que variable de chaîne

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de VARIABLE.

7.39.3 Spécification d'une chaîne en tant que valeur d'énumération

Objet

Une chaîne de valeur d'énumération est une chaîne associée à une des valeurs d'une variable d'énumération.

Structure lexicale

`reference, value`

Les attributs sont spécifiés dans le Tableau 245.

Tableau 245 – Chaîne en tant que valeur d'énumération

Utilisa-tion	Attribut	Description
m	reference	Référence à une instance de VARIABLE de TYPE ENUMERATED ou BIT_ENUMERATED.
m	value	Valeur correspondante au texte de description dans la liste d'énumération.

EXEMPLE 1 La chaîne de valeur d'énumération suivante spécifie la chaîne associée à la valeur 4 de la variable ENUMERATED units_code: units_code (4).

EXEMPLE 2 La chaîne de valeur d'énumération suivante spécifie la chaîne associée à la valeur 0x02 de la variable BIT_ENUMERATED var_bitenum: var_bitenum (0x02).

7.39.4 Spécification d'une chaîne en tant que référence de dictionnaire

Objet

Une référence de dictionnaire pointe vers une chaîne dans un dictionnaire de texte.

Structure lexicale

reference

L'attribut est spécifié dans le Tableau 246.

Tableau 246 – Chaîne en tant que référence de dictionnaire

Utilisa-tion	Attribut	Description
m	reference	Référence à une ou plusieurs chaînes spécifiées dans un dictionnaire (voir 7.41).

7.39.5 Référencement d'attributs HELP et LABEL d'instances d'EDD

Objet

Pour obtenir la chaîne actuelle de LABEL ou HELP d'une instance d'EDD, l'instance et l'attribut LABEL ou HELP doivent être référencés.

Structure lexicale

reference, [HELP, LABEL]

Les attributs sont spécifiés dans le Tableau 247.

Tableau 247 – Référencement des attributs HELP et LABEL des instances d'EDD

Utilisa-tion	Attribut	Description
m	reference	Instance d'EDD qui contient les attributs LABEL et/ou HELP.
s	HELP	Chaîne HELP de l'instance.
s	LABEL	Chaîne LABEL de l'instance.

7.39.6 Opérations de chaîne

Objet

Les littéraux de chaîne, les variables de chaîne, les valeurs d'énumération chaîne, les références de dictionnaire et les valeurs d'attribut de constructions de base peuvent être concaténés les uns avec les autres. Si un terme contient des références de types de données numériques, les valeurs doivent être converties en chaîne par l'application EDD.

EXEMPLE `variable_of_type_string = "Description"`
`"Electronic" + "Device" + variable_of_type_string` conduit à `"Electronic Device Description"`.

Structure lexicale

`(string)<exp>, [(concatenation-operator, string)<exp>]+`

Les attributs sont spécifiés dans le Tableau 248.

Tableau 248 – Opération de chaîne

Utilisa-tion	Attribut	Description
m	string	Littéraux de chaîne, variables de type de données de chaîne ou numériques, valeurs d'énumération de chaîne, références de dictionnaire ou références à des attributs de chaîne, par exemple HELP ou LABEL. L'application EDD doit convertir les expressions évaluées et le type de données numériques en chaînes (voir Article A.3).
o	concatenation-operator	Les chaînes doivent être concaténées.

7.39.7 Formats de chaîne d'invite de commande

Objet

Des chaînes d'invite de commande peuvent être utilisées pour imbriquer des éléments formatés dans des chaînes. Ces éléments sont composés d'un spécificateur de format facultatif et d'un attribut VARIABLE ID ou d'une référence.

Structure lexicale

`format, (reference, id)`
`label, (reference, id)`

Les attributs sont spécifiés dans le Tableau 249.

Tableau 249 – Spécificateur de format

Utilisa-tion	Attribut	Description
m	format	Chaîne du format printf du langage de programmation C (voir ISO/IEC 9899). Si le format n'est pas spécifié, le format de l'instance de VARIABLE référencée est utilisé. Si aucun format n'est spécifié, un format par défaut est utilisé sur la base du type d'instance de VARIABLE.
m	label	Attribut LABEL de l'instance de VARIABLE.
o/s	reference	Valeur ou étiquette de l'instance de VARIABLE référencée.
o/s	id	ID d'élément des attributs VARIABLE.

7.40 Expression

7.40.1 Structure générale

Objet

Il existe trois types d'expressions.

- Les expressions primaires comprennent des constantes, des expressions entre parenthèses, des références de VARIABLE, des accès d'élément d'index ou un accès au statut d'instances de VARIABLE. Ces opérations sont associatives par la gauche, c'est-à-dire qu'elles sont évaluées de gauche à droite.
- Des expressions unaires comprennent tous les opérateurs évaluant un opérande unique. Ces opérations sont associatives par la droite, c'est-à-dire qu'elles sont évaluées de droite à gauche.
- Les expressions binaires comprennent tous les opérateurs évaluant deux opérandes. Ces opérations sont associatives par la gauche, c'est-à-dire qu'elles sont évaluées de gauche à droite.

7.40.2 Expressions primaires

Le Tableau 250 ci-dessous résume les expressions primaires dans le langage EDDL.

Tableau 250 – Expressions primaires

Expression primaire	Description
constante	Expression dont la valeur est identique à la valeur de la constante.
expression entre parenthèses	Expression dont la valeur est identique à la valeur de l'expression incluse.
référence de variable	Expression dont la valeur est la valeur de l'attribut VARIABLE référencé.
ARRAY_INDEX	Expression dont la valeur est dérivée du numéro d'index de l'attribut VALUE_ARRAY ou LIST référençant.
valeurs d'attributs VARIABLE	Référence constituée de l'identificateur d'une instance de VARIABLE et d'un attribut. La référence fournit la valeur correspondante de l'attribut (voir Tableau 251 pour les valeurs retournées).
PI	Constante mathématique π en tant que double.
valeurs d'un attribut AXIS	Référence à une instance et un attribut AXIS. La référence fournit la valeur correspondante de l'attribut (voir Tableau 252 pour les valeurs retournées). Ces attributs peuvent uniquement être lus, sauf indication contraire.
valeurs d'un attribut BLOB	Référence à une instance et un attribut BLOB. La référence fournit la valeur correspondante de l'attribut (voir Tableau 253 pour les valeurs retournées). Ces attributs peuvent uniquement être lus, sauf indication contraire.
valeurs d'attribut LIST	Référence à une instance et un attribut LIST. La référence fournit la valeur correspondante de l'attribut (voir Tableau 254 pour les valeurs retournées). Ces attributs peuvent uniquement être lus, sauf indication contraire.
valeurs d'attribut ARRAY	Référence à une instance et un attribut ARRAY. La référence fournit la valeur correspondante de l'attribut (voir Tableau 2545 pour les valeurs retournées). Ces attributs peuvent uniquement être lus, sauf indication contraire.
LINEAR	Echelle linéaire.
LOGARITHMIC	Echelle logarithmique.
TRUE	Valeur booléenne.
FALSE	Valeur booléenne.
NULL	Objet ou élément inconnu.

Expression primaire	Description
FIRST	Pour le premier élément LIST de la liste. Pour OBJECT_REFERENCE, premier objet parent dans la hiérarchie d'objet.
LAST	Pour le dernier élément LIST de la liste. Pour OBJECT_REFERENCE, navigation jusqu'au dernier objet parent dans la hiérarchie d'objet.
CHILD	Pour OBJECT_REFERENCE, navigation jusqu'au premier objet descendant.
PARENT	Pour OBJECT_REFERENCE, navigation jusqu'au niveau de hiérarchie supérieur suivant.
NEXT	Pour OBJECT_REFERENCE, navigation jusqu'à l'objet parent suivant.
PREV	Pour OBJECT_REFERENCE, navigation jusqu'à l'objet parent précédent.
SELF	Pour OBJECT_REFERENCE, navigation jusqu'au propre objet ("ce").

IECNORM.COM :: Click to view the full PDF of IEC 61804-3:2015

Tableau 251 – Valeurs d'attribut VARIABLE

Attributs	Type de données	Description
MIN_VALUE	type d'instance de VARIABLE	Valeur qui est conforme à l'attribut MIN_VALUE de l'instance de VARIABLE. Si plus d'un attribut MIN_VALUE est défini, MIN_VALUE et l'identificateur d'intervalle (voir Tableau 168) doivent être utilisés pour le référencement. Si aucun attribut MIN_VALUE n'est défini, il convient que la valeur minimale spécifique du type de donnée (voir Tableau D.9) soit utilisée dans l'évaluation.
MAX_VALUE	type d'instance de VARIABLE	Valeur qui est conforme à l'attribut MAX_VALUE de l'instance de VARIABLE. Si plus d'un attribut MAX_VALUE est défini, MAX_VALUE et l'identificateur d'intervalle (voir Tableau 168) doivent être utilisés pour le référencement. Si aucun attribut MAX_VALUE n'est défini, il convient que la valeur maximale spécifique du type de donnée (voir Tableau D.9) soit utilisée dans l'évaluation.
SCALING_FACTOR	double	Valeur qui est conforme à l'attribut SCALING_FACTOR de l'instance de VARIABLE.
DEFAULT_VALUE	type d'instance de VARIABLE	Valeur qui est conforme à l'attribut DEFAULT_VALUE de l'instance de VARIABLE. Si aucun attribut DEFAULT_VALUE n'est défini, il convient que la valeur initiale spécifique du type de donnée (voir Tableau D.9) soit utilisée dans l'évaluation. Dans des expressions avec DEFAULT_VALUE, l'attribut variable est toujours référencé, même si COMPONENT_INITIAL_VALUES ou TEMPLATE_DEFAULT_VALUES est défini.
INITIAL_VALUE	type d'instance de VARIABLE	Valeur qui est conforme à l'attribut INITIAL_VALUE de l'instance de VARIABLE. Si aucun attribut INITIAL_VALUE n'est défini, il convient que la valeur initiale spécifique du type de donnée (voir Tableau D.9) soit utilisée dans l'évaluation.
VARIABLE_STATUS		Valeur qui indique le statut de VARIABLE. Les statuts suivants sont définis: ▪ VARIABLE_STATUS_NONE spécifie que la valeur est valide et non un des autres statuts; ▪ VARIABLE_STATUS_INVALID spécifie que la valeur est hors de l'intervalle; ▪ VARIABLE_STATUS_NOT_ACCEPTED spécifie que la valeur n'a pas pu être transférée; ▪ VARIABLE_STATUS_NOT_SUPPORTED spécifie que la valeur n'est pas prise en charge; ▪ VARIABLE_STATUS_CHANGED spécifie que la valeur a été modifiée par l'utilisateur ou par une méthode; ▪ VARIABLE_STATUS_LOADED spécifie que la valeur a été chargée avec succès; ▪ VARIABLE_STATUS_INITIAL spécifie que la valeur est inchangée; ▪ VARIABLE_STATUS_NOT_CONVERTED spécifie que la valeur n'a pas été convertie, par exemple, parce que les unités ne sont pas compatibles.

Tableau 252 – Valeurs d'attribut AXIS

Attributs	Type de données	Description
MIN_VALUE	type arithmétique	Valeur qui est conforme à l'attribut MIN_VALUE de l'instance d'AXIS.
MAX_VALUE	type arithmétique	Valeur qui est conforme à l'attribut MAX_VALUE de l'instance d'AXIS.
SCALING	LINEAR ou LOGARITHMIC	Valeur qui est conforme à l'attribut SCALING de l'instance d'AXIS.
VIEW_MIN	type arithmétique	Valeur qui est conforme à l'attribut VIEW_MIN de l'instance d'AXIS; VIEW_MIN peut être lu et écrit.
VIEW_MAX	type arithmétique	Valeur qui est conforme à l'attribut VIEW_MAX de l'instance d'AXIS; VIEW_MAX peut être lu et écrit.

Tableau 253 – Valeurs d'attribut BLOB

Attributs	Type de données	Description
COUNT	unsigned_integer	Valeur qui est conforme à l'attribut COUNT de l'instance de BLOB. La taille de BLOB est retournée, c'est-à-dire le nombre d'octets.
IDENTITY	string	Valeur qui est conforme à l'attribut IDENTITY de l'instance de BLOB. La valeur sera une chaîne vide si l'attribut IDENTITY n'est pas spécifié et si le Builtin d'identité n'a pas été appelé.

Tableau 254 – Valeurs d'attribut LIST

Attributs	Type de données	Description
CAPACITY	unsigned_integer	Valeur qui est conforme à l'attribut CAPACITY de l'instance de LIST.
COUNT	unsigned_integer	Valeur qui est conforme à l'attribut CAPACITY de l'instance de LIST.

Tableau 255 – Valeurs d'attribut ARRAY

Attributs	Type de données	Description
NUMBER_OF_ELEMENTS	integer	Valeur qui est conforme à l'attribut NUMBER_OF_ELEMENTS de l'instance de VALUE_ARRAY.

7.40.3 Expressions unaires

Une expression unaire est constituée d'un opérande numérique, d'une expression, précédée par un opérateur unaire. Le Tableau 256 spécifie les expressions unaires dans l'EDDL.

Tableau 256 – Expressions unaires

Opérateur	Description
-	Négation arithmétique de l'opérande.
~	Négation binaire de l'opérande, c'est-à-dire que chaque bit du résultat est l'inverse du bit correspondant de l'opérande. L'opérande de l'opérateur ~ doit avoir une valeur entière.
!	Négation logique de l'opérande.

7.40.4 Expressions binaires

7.40.4.1 Structure générale

Une expression binaire est constituée de deux opérandes ou expressions, séparés par un opérateur binaire. Si un opérande a une valeur à virgule flottante, l'autre opérande est converti (promu) en valeur à virgule flottante. Le paragraphe 7.40.4 spécifie les types d'expressions binaires suivants:

- Multiplicatif
- Additif
- Décalage
- Relationnel
- Egalité
- ET binaire (&)
- XOUI binaire (^)
- OU binaire (|)
- ET logique (&&)
- OU logique (||)
- Evaluation conditionnelle

7.40.4.2 Opérateurs multiplicatifs

Les opérateurs multiplicatifs spécifient la multiplication et la division d'opérandes. Le Tableau 257 spécifie les opérateurs multiplicatifs.

Tableau 257 – Opérateurs multiplicatifs

Opérateur	Description
*	Multiplication de ses opérandes.
/	Division du premier opérande par le second opérande. Le résultat de l'opérateur / est le quotient de la division.
%	Division du premier opérande par le second opérande. Le résultat de l'opérateur % est le résidu.

7.40.4.3 Opérateurs additifs

Les opérateurs additifs spécifient l'addition et la soustraction d'opérandes numériques. Si cet opérateur est appliqué à des opérandes numériques, le résultat est alors la somme de ses deux opérandes, et l'opération est commutative. Si l'opérateur + est appliqué à des opérandes chaînes, le résultat est une valeur de chaîne générée par ajout d'une seconde chaîne à la première chaîne et l'opération n'est pas commutative.

L'opérateur moins indique une soustraction. La valeur résultante de cette opération est la différence entre ses opérandes. Le second opérande est soustrait du premier. Le Tableau 258 spécifie les opérateurs additifs.

Tableau 258 – Opérateurs additifs

Opérateur	Description
+	Addition de ses opérandes.
-	Soustraction du second opérande du premier.

7.40.4.4 Opérateurs de décalage

Les opérateurs << et >> spécifient un décalage du premier opérande du nombre de bits spécifié par le second opérande. Les opérandes des opérateurs << et >> doivent avoir des valeurs entières. Le Tableau 259 spécifie les opérateurs de décalage.

Tableau 259 – Opérateurs de décalage

Opérateur	Description
<<	Premier opérande sur la gauche. Les bits décalés à l'extérieur sont rejetés, et les bits laissés vacants sont remplis avec des zéros.
>>	Premier opérande sur la droite. Les bits décalés à l'extérieur sont rejetés. Si le premier opérande est inférieur à 0, les bits laissés vacants sont remplis avec des 1, sinon ils sont remplis avec des zéros.

7.40.4.5 Opérateurs relationnels

Les opérateurs relationnels (<, <=, >, >=) spécifient une comparaison des opérandes. Le résultat de ce type d'expression est 1 si la relation soumise à l'essai est vraie; sinon, le résultat est 0. Les opérandes doivent constituer un type de données numériques. Le Tableau 260 spécifie les opérateurs relationnels.

Tableau 260 – Opérateurs relationnels

Opérateur	Description
<	Soumet à l'essai la relation "inférieur à".
<=	Soumet à l'essai la relation "inférieur ou égal à".
>	Soumet à l'essai la relation "supérieur à".
>=	Soumet à l'essai la relation "supérieur ou égal à".

7.40.4.6 Opérateurs d'égalité

Les opérateurs d'égalité sont == et !=. Le résultat de ce type d'expression est 1 si la relation soumise à l'essai est vraie; sinon, le résultat est 0. Les opérateurs doivent appartenir au même type de données. Tous les types de données sont admis. Le Tableau 261 spécifie les opérateurs d'égalité.

Tableau 261 – Opérateurs d'égalité

Opérateur	Description
==	Soumet à l'essai la relation "est égal à".
!=	Soumet à l'essai la relation "est différent de".

7.40.4.7 Opérateur ET binaire (&)

L'opérateur & spécifie le ET binaire de ses opérandes entiers, c'est-à-dire que chaque bit du résultat est défini si chacun des bits correspondants des opérandes est défini. Les opérandes de l'opérateur & doivent avoir des valeurs entières.

7.40.4.8 Opérateur XOUI binaire (^)

L'opérateur ^ spécifie le OU exclusif binaire de ses opérandes entiers, c'est-à-dire que chaque bit du résultat est défini si un seul des bits correspondants des opérandes est défini. Les opérandes de l'opérateur ^ doivent avoir des valeurs entières.

7.40.4.9 Opérateur OU binaire (|)

L'opérateur | spécifie le OU inclusif binaire de ses opérandes entiers, c'est-à-dire que chaque bit du résultat est défini si l'un des bits correspondants des opérandes est défini. Les opérandes de l'opérateur | doivent avoir des valeurs entières.

7.40.4.10 Opérateur ET logique (&&)

L'opérateur && spécifie l'évaluation ET booléenne de ses opérandes entiers. Si l'un des opérandes référence un attribut VARIABLE non valide (dont VALIDITY est égal à FALSE), l'opérande doit être égal à 0. Le résultat de ce type d'expression est 1 si les deux opérandes sont différents de 0; sinon, le résultat est 0. Si le premier opérande est égal à 0, le second opérande n'est pas évalué.

7.40.4.11 Opérateur OU logique (||)

L'opérateur || spécifie l'évaluation OU booléenne de ses opérandes entiers. Si l'un des opérandes référence un attribut VARIABLE non valide (dont VALIDITY est égal à FALSE), l'opérande doit être égal à 0. Le résultat de ce type d'expression est 1 si l'un des opérandes est différent de 0; sinon, le résultat est 0. Si le premier opérande est différent de 0, le second opérande n'est pas évalué.

7.40.4.12 Evaluation conditionnelle

Les opérateurs ? et: sont utilisés pour l'évaluation conditionnelle.

EXEMPLE 1

```
MIN_VALUE (var1 > 0) ? 10: 0;
```

L'expression logique est évaluée et si elle est différente de zéro, l'attribut MIN_VALUE est 10; sinon, il est égal à 0.

EXEMPLE 2 La structure if-else suivante a une logique identique à l'exemple 1:

```
MIN_VALUE IF (var1 > 0) { 10; } ELSE { 0; }
```

7.41 Dictionnaire de texte

Structure générale

Le dictionnaire est une collection de chaînes de textes qui peuvent être utilisées dans une EDD. Le texte du dictionnaire est référencé dans une EDD. Le dictionnaire peut être consulté depuis n'importe quelle EDD afin d'assurer la cohérence des chaînes de texte (par exemple, pour utilisation dans des attributs LABEL et HELP).

Généralement, un dictionnaire de texte commun est utilisé pour un groupe d'appareils types. De plus, un dictionnaire spécifique pour l'utilisateur peut être spécifié. Le dictionnaire de texte prend en charge des textes multilingues.

Structure lexicale

```
section-number, string-number identifier (language-code, country-code, string)+
```

Les attributs sont spécifiés dans le Tableau 262.

Tableau 262 – Attributs de dictionnaire de texte

Utilisa-tion	Attribut	Description
m	section-number	Définition d'une section dans un tableau de code. Si aucun tableau de code n'est utilisé, cette valeur est '0'.
m	string-number	Définition d'un décalage de l'entrée de dictionnaire dans un tableau de code. Si aucun tableau de code n'est utilisé, cette valeur est '0'.
m	identifiant	Nom d'une chaîne de texte dans une EDD. L'identificateur doit être unique dans une combinaison de section-number et de string-number pour tous les dictionnaires utilisés. Les identificateurs doivent être uniques à l'exception de xxx_highoffset pour terminer une section.
o	language-code	Langue utilisée et présentation ASCII ou multicode d'une chaîne; les codes de langue sont définis dans l'ISO 639 (voir A.2.4).
o	country-code	Pays dans lequel la langue est parlée; les codes de pays sont définis dans l'ISO 3166-1. Le code de pays ne peut être spécifié que si le code de langue est défini.
m	string	Chaîne de texte dans la langue sélectionnée.

Les littéraux de chaîne sont définis en A.2.4.

7.42 PLUGIN

7.42.1 Structure générale

Objet

PLUGIN décrit un plug-in d'interface utilisateur (UIP, User Interface Plug-in).

Structure lexicale

PLUGIN identifiant [HELP, LABEL, UUID, VALIDITY, VISIBILITY]

Les attributs PLUGIN sont spécifiés dans le Tableau 263.

Tableau 263 – Attributs PLUGIN

Utilisa-tion	Attribut	Description
m	LABEL	Elément lexical (voir 7.36.9).
m	UUID	Elément lexical (voir 7.42.2).
o	HELP	Elément lexical (voir 7.36.8).
o	VALIDITY	Elément lexical (voir 7.36.16).
o	VISIBILITY	Elément lexical (voir 7.36.19).

7.42.2 Attribut spécifique – UUID

Objet

L'attribut UUID spécifie l'identificateur unique utilisé pour identifier le PLUGIN. L'attribut UUID est un identificateur universel unique tel que spécifié dans la norme ISO/IEC 9834-8.

NOTE Cet attribut peut ne pas être conditionnel.

Structure lexicale

UUID uuid

L'attribut UUID est spécifié dans le Tableau 264.

Tableau 264 – Attribut UUID

Utilisa-tion	Attribut	Description
m	uuid	Identificateur unique du plug-in spécifié dans le manifeste.

7.43 BLOB

Objet

Le BLOB (Binary Large Object) doit être utilisé pour transférer des données binaires de ou vers un appareil. L'application EDD a un accès en lecture seule au contenu d'un attribut BLOB, à des fins de validation uniquement par exemple. Un BLOB est un conteneur de données dont le contenu est accessible via des Builtins.

Structure lexicale

BLOB identifier [HANDLING, HELP, IDENTITY, LABEL]

Les attributs BLOB sont spécifiés dans le Tableau 265.

Tableau 265 – Attributs BLOB

Utilisa-tion	Attribut	Description
m	HANDLING	Élément lexical (voir 7.36.6). NOTE HANDLING READ signifie que l'attribut BLOB ne peut être lu que depuis l'appareil.
o	HELP	Élément lexical (voir 7.36.8).
o	IDENTITY	Élément lexical (voir 7.36.21). Si IDENTITY n'est pas défini, le nom de fichier doit être sélectionné via une méthode (voir IEC 61804-5).
o	LABEL	Élément lexical (voir 7.36.9).

Annexe A (normative)

Définition formelle d'EDDL

A.1 Préprocesseur EDDL

A.1.1 Structure générale

Objet

Le préprocesseur génère une EDD avant compilation. Le préprocesseur remplace des parties du texte d'EDD, insère le contenu d'autres fichiers ou supprime le traitement des parties de l'EDD en supprimant des sections.

Structure

Toutes les directives du préprocesseur démarrent avec un symbole dièse (#) comme premier caractère sur une ligne. Une espace (caractères ESPACE ou TAB) peut apparaître après le # initial pour obtenir une indentation correcte.

A.1.2 Directives

A.1.2.1 #define

Objet

La directive #define donne un nom significatif à une constante dans l'EDD.

Structure

```
#define identifieur token-string
```

Cette forme de directive #define donne l'instruction au préprocesseur de remplacer toutes les occurrences suivantes de l'identificateur dans le fichier de l'EDD par token-string.

NOTE L'identificateur n'est pas remplacé s'il apparaît dans un commentaire, dans une chaîne ou dans un autre identificateur plus long.

```
#define identifieur
```

Cette forme de directive #define donne l'instruction au préprocesseur de supprimer toutes les occurrences de l'identificateur dans le fichier EDD. L'identificateur reste défini et peut être soumis à l'essai en utilisant les directives #if defined et #ifdef.

```
#define identifieur (argument, ... ,argument) token-string
```

Cette forme de la directive #define donne l'instruction au préprocesseur de créer des macros de type fonction. Cette forme accepte une liste d'arguments qui doivent apparaître entre parenthèses et sont séparés par des virgules.

Lorsqu'une macro a été définie sous cette forme de syntaxe, les instances textuelles consécutives suivies par une liste d'arguments constituent un appel de macro. Les arguments réels suivant une instance d'identificateur dans le fichier source sont mis en correspondance avec les paramètres formels correspondants dans la définition de la macro. Chaque paramètre formel dans token-string est remplacé par l'argument réel correspondant. Toutes les macros dans l'argument réel sont développées avant que la directive ne remplace le paramètre formel.

Chaque argument dans la liste doit être unique. Aucun espace n'est admis pour séparer un identificateur et la parenthèse d'ouverture.
Pour les directives longues sur des lignes sources multiples, une barre oblique inverse (\) doit être utilisée avant le caractère NEWLINE.
Aucun espace n'est admis entre le nom et "(".

A.1.2.2 **#include**

Objet

La directive `#include` indique au préprocesseur de traiter le contenu d'un fichier spécifié comme si ce contenu était apparu dans le programme source au point où la directive apparaît.

Structure

```
#include "filename"  
#include <filename>
```

Lire le contenu du fichier nommé avec le nom de fichier. Le préprocesseur traite ces données comme si elles faisaient partie du fichier actuel.

Les deux formes de syntaxe causent le remplacement de cette directive par le contenu entier du fichier inclus spécifié. La différence entre les deux formes est l'ordre dans lequel le préprocesseur recherche les fichiers inclus lorsque le chemin n'est pas complètement spécifié.

Aucun jeton supplémentaire ne peut se trouver sur la ligne de directive après le " ou > final.

A.1.2.3 **#line**

Objet

La directive `#line` conduit le préprocesseur à remplacer le numéro de ligne et le nom de fichier stockés en interne par le compilateur par un numéro de ligne et un nom de fichier donnés. Le processeur de l'EDD utilise le numéro de ligne et le nom de fichier pour faire référence aux erreurs qu'il rencontre pendant le traitement. Le numéro de ligne fait généralement référence à la ligne d'entrée actuelle, et le nom du fichier fait référence au fichier d'entrée actuel. Le numéro de ligne est incrémenté une fois que chaque ligne est traitée.

Structure

```
#line integer-constant "filename"
```

`integer-constant` et `filename` conduisent le préprocesseur à remplacer le numéro de ligne stocké en interne par le processeur de l'EDD par un numéro de ligne et un nom de fichier donnés. `integer-constant` est interprété comme étant le numéro de ligne de la ligne suivante et est incrémenté une fois que chaque ligne est traitée.

A.1.2.4 **#if, #elif, #else, et #endif**

Objet

La directive `#if`, avec les directives `#elif`, `#else` et `#endif`, contrôle le traitement des parties d'un fichier EDD.

```
#if constant-expression
```

Les lignes suivantes jusqu'à la directive `#else`, `#elif` ou `#endif`, apparaissent dans la sortie uniquement si `constant-expression` produit une valeur différente de zéro. Les opérateurs

binaires "&&" et "||" sont admis dans constant-expression ainsi que l'opérateur "?:" et les opérateurs unaires "-", "!" et "~".

De plus, l'opérateur unaire défini peut être utilisé dans des expressions constantes spéciales, comme décrit par la syntaxe suivante:

```
defined(identifieur)
defined identifieur
```

Cette expression constante est considérée comme étant vraie (différente de zéro) si l'identificateur est actuellement défini; sinon, la condition est fausse. Un identificateur défini comme un texte vide est considéré comme étant défini. La directive définie peut être utilisée dans une directive #if et #elif, mais pas ailleurs.

```
#elif constant-expression
```

Un nombre quelconque de directives #elif peuvent apparaître entre une directive #if, #ifdef ou #ifndef et une directive #else ou #endif correspondante. Les lignes suivant la directive #elif apparaissent dans la sortie uniquement si toutes les conditions suivantes sont vraies:

- La constant-expression dans les directives #if précédentes est évaluée à zéro, le nom dans le #ifdef précédent n'est pas défini ou le nom dans les directives #ifndef précédentes est défini.
- Les constant-expression (expressions constantes) dans toutes les directives #elif inversées sont évaluées à zéro.
- La constant-expression actuelle est évaluée à une valeur différente de zéro.

Si la constant-expression est évaluée à une valeur différente de zéro, ignorer les directives #elif et #else suivantes jusqu'à la directive #endif correspondante. Toute constant-expression admise dans une directive #if l'est également dans une directive #elif.

```
#else
```

Si toutes les occurrences de constant-expression sont fausses, ou si aucune directive #elif n'apparaît, le préprocesseur sélectionne le bloc de texte après l'article #else. Si l'article #else est omis et que toutes les instances de constant-expression dans le bloc #if sont fausses, aucun bloc de texte n'est sélectionné.

```
#endif
```

Termine une section de lignes qui commence avec une des directives conditionnelles #if, #ifdef ou #ifndef. Chaque directive doit correspondre à un #endif.

NOTE 1 Le préprocesseur reconnaît des paramètres formels pour des macros dans des corps de directive #define, même lorsqu'ils se trouvent à l'extérieur de constantes de caractère et de chaînes entre guillemets. Les noms de macro ne sont pas reconnus dans des constantes de caractère ou des chaînes entre guillemets pendant une analyse normale. Par conséquent, #define aaa bbb ne développe pas aaa en LABEL "aaa".

NOTE 2 Les macros ne sont pas développées en traitant une directive #define ou #undef. Par conséquent:

```
#define aaa bbb

#define ccc aaa

#undef aaa

ccc

produit aaa.
```

NOTE 3 Les macros ne sont pas développées pendant l'analyse qui détermine les paramètres réels pour un autre appel de macro.

```
Par conséquent:

#define turn(aaa,bbb) bbb aaa
```

```
#define ccc ddd

turn(ccc, #define ccc eee)

produit #define ddd eee ddd.
```

A.1.2.5 #ifdef, #ifndef et #undef

Objet

Les directives `#ifdef` et `#ifndef` contrôlent le traitement des parties d'un fichier EDD. Les directives `#ifdef` et `#ifndef` effectuent la même tâche que la directive `#if` lorsqu'elle est utilisée avec un identificateur défini.

```
#ifdef identifier
```

Les lignes suivantes jusqu'à la directive `#else`, `#elif` ou `#endif` correspondante apparaissent dans la sortie uniquement si l'identificateur a été défini. L'identificateur est spécifié en utilisant une directive `#define` ou à l'extérieur de l'EDD et en l'absence d'une directive `#undef` intercalée. Aucun jeton additionnel ne peut se trouver sur la ligne de directive après le nom.

```
#ifndef identifier
```

Les lignes suivantes jusqu'à la directive `#else`, `#elif` ou `#endif` correspondante apparaissent dans la sortie uniquement si le nom n'a pas été défini ou si sa définition a été supprimée avec une directive `#undef`. Aucun jeton additionnel ne peut se trouver sur la ligne de directive après le nom.

```
#undef identifier
```

La directive `#undef` cause la suppression de la définition d'un identificateur par le préprocesseur. Aucun jeton additionnel ne peut se trouver sur la ligne de directive après le nom.

A.1.3 Macros prédéfinies

A.1.3.1 Structure générale

Le préprocesseur doit reconnaître un ensemble spécifié de macros prédéfinies. Il convient que ces macros soient utilisées ailleurs dans l'EDD. Elles sont indiquées à titre d'information.

Ces macros n'acceptent aucun argument et ne peuvent pas être redéfinies.

A.1.3.2 Liste de macros prédéfinies

A.1.3.2.1 __FILE__

Objet

`__FILE__` contient le nom du fichier EDD actuel.

Structure

```
__FILE__
```

`__FILE__` est développé en chaîne entourée par des guillemets doubles.

A.1.3.2.2 __LINE__

Objet

`__LINE__` contient le numéro de ligne dans le fichier EDD actuel.

Structure

__LINE__

__LINE__ est une constante entière décimale. Elle peut être modifiée avec une directive #line.

A.1.4 Caractères NEWLINE

Un caractère NEWLINE termine une constante de caractère ou une chaîne entre guillemets. La barre oblique inverse (\) suivie d'un attribut NEWLINE dans le corps d'une instruction #define est utilisée pour continuer la définition sur la ligne suivante.

A.1.5 Commentaires

Un commentaire est une séquence de caractères commençant par une combinaison barre oblique/astérisque (/*) et se termine par une combinaison astérisque/barre oblique (*). Un commentaire peut comprendre une combinaison quelconque de caractères de l'ensemble de caractères représentable, comprenant des caractères NEWLINE.

De plus, le commentaire à une ligne est pris en charge s'il est précédé par deux barres obliques (//). Un commentaire commençant par deux barres obliques est terminé par le caractère NEWLINE suivant qui n'est pas précédé par un caractère d'échappement.

A.2 Conventions

A.2.1 Constantes entières

Une constante entière peut être spécifiée en notation binaire, octale, décimale ou hexadécimale. Le Tableau A.1 spécifie les conventions pour chaque type de notation.

Tableau A.1 – Conventions pour les constantes entières

Type de constante entière	Conventions
binaires	Séquence non vide de chiffres binaires 0 et 1 précédée par 0b ou 0B.
octales	Séquence non vide des chiffres 0 à 7 commençant par 0.
décimales	Séquence non vide des chiffres décimaux 0 à 9, ne commençant pas par 0.
hexadécimales	Séquence non vide de chiffres hexadécimaux précédée par 0x ou 0X. Les chiffres hexadécimaux sont les chiffres 0 à 9 et les lettres a à f (ou A à F) avec les valeurs 10 à 15, respectivement.

A.2.2 Constantes à virgule flottante

Structure lexicale

Une constante à virgule flottante comprend quatre parties:

- une partie entière, une séquence de chiffres décimaux;
- un point décimal (.);
- une partie fractionnaire, une séquence de chiffres décimaux;
- une partie exposant, une séquence éventuellement signée de chiffres décimaux précédée par une des lettres e ou E.

Règles

Les règles suivantes s'appliquent à l'utilisation de constantes à virgule flottante:

- la partie entière ou la partie fractionnaire peut être omise, mais pas les deux;
- le point décimal ou la partie exposant peut être omis(e), mais pas les deux.

Exemple: Des exemples de constantes à virgule flottante sont présentés ci-dessous:

59,
 ,87
 48,93
 4,8e12

Le calcul de l'expression algébrique doit utiliser le type de données le plus précis.

A.2.3 Littéraux de chaîne

Les littéraux de chaîne dans une EDD doivent être codés avec ISO Latin-1 (ISO/IEC 8859-1) ou UTF-8 (RFC 3629). Dans une EDD donnée, tous les littéraux de chaîne doivent utiliser le même codage. Un littéral de chaîne est une séquence de caractères éventuellement vide entourée par des guillemets doubles ("). Les caractères entre guillemets doivent être un caractère quelconque sauf les suivants:

- guillemet (");
- barre oblique (\);
- saut de ligne.

Avec les séquences d'échappement dans les littéraux de chaîne, une chaîne constante peut également contenir des séquences d'échappement qui représentent un caractère. Le Tableau A.2 présente les séquences d'échappement et leur résultat.

Tableau A.2 – Utilisation de séquences d'échappement dans des littéraux de chaîne

Code d'échappement	Résultat
'	Guillemet simple.
"	Guillemet double.
	Barre verticale.
?	Point d'interrogation.
\	Barre oblique inversée.
\a	Alerte.
\f	Saut de page.
\n	Saut de ligne.
\r	Saut de paragraphe.
\t	Tabulation horizontale.
\v	Tabulation verticale.

A.2.4 Utilisation de codes de langue et de pays dans des littéraux de chaîne

Un littéral de chaîne peut encapsuler toutes les traductions d'une phrase donnée. Le code de langue est spécifié à l'aide de l'ISO/IEC 8859-1. Si un littéral de chaîne ne contient pas de traductions pour toutes les langues, une langue par défaut doit être utilisée. Le Tableau A.3 montre des exemples de codes de langue.

Tableau A.3 – Exemples de codes de langue pour les littéraux de chaîne

Code de langue	Langue
Aucun code assigné	Langue par défaut, utilisée si une langue est demandée et n'est pas définie.
zh	Chinois
de	Allemand
en	Anglais
es	Espagnol
fr	Français
it	Italien
ja	Japonais
ko	Coréen

L'anglais doit être la langue par défaut.

EXEMPLE 1 Le littéral de chaîne suivant spécifie l'expression anglaise "Invalid selection" en anglais et en allemand:

"Invalid Selection"

"|de|Unzulässige Auswahl"

En plus des codes de pays normalisés définis dans l'ISO 3166-1, le code pays zz peut être utilisé pour spécifier une version courte d'une chaîne. Des versions courtes de chaînes sont utiles pour les applications EDD exécutées sur des appareils mobiles qui ont une taille d'écran limitée.

EXEMPLE 2

write_protected

"Write Protected"

"|de|Schreibgeschützt"

"|fr|Protégé en écriture"

"|it|Protetto in scrittura"

"|es|Protegido contra escritura"

"|en zz|Wr Prot"

"|de zz|Geschützt"

L'identificateur write_protected est l'identificateur qui est référencé dans l'EDD. Les qualificatifs |de|, |fr|, |it|, |es| spécifient le texte pour les langues Allemand, Français, Italien et Espagnol. Les qualificatifs "en zz" et "de zz" spécifient des versions courtes des chaînes anglaise et allemande. La même approche peut également être utilisée pour fournir des versions courtes pour le français, l'italien et l'espagnol.

A.3 Opérateurs

Le Tableau A.4 définit la priorité et l'associativité pour les opérateurs EDDL.

Tableau A.4 – Priorité et associativité pour les opérateurs EDDL

Priorité	Associativité	Opérateur	Prise en charge en dehors des attributs METHOD	Description	Catégorie
1 (la plus élevée)	De gauche à droite	++	Non	Incrément de suffixe	Arithmétique
1	De gauche à droite	--	Non	Décrément de suffixe	Arithmétique
1	De gauche à droite	()	Oui	Appel de fonction	Appel de fonction
1	De gauche à droite	[]	Oui	Indexation de matrice	Référencement
1	De gauche à droite	.	Oui	Référencement de sous-éléments et	Référencement
1	De gauche à droite	->	Oui	Référencement d'objet	Référencement
2	De droite à gauche	++	Non	Incrément de préfixe	Arithmétique
2	De droite à gauche	--	Non	Décrément de préfixe	Arithmétique
2	De droite à gauche	+	Oui	Plus unaire	Arithmétique
2	De droite à gauche	-	Oui	Moins unaire	Arithmétique
2	De droite à gauche	!	Oui	NON logique	Arithmétique
2	De droite à gauche	~	Oui	NON binaire	Binaire
3	De gauche à droite	*	Oui	Multiplication arithmétique	Arithmétique
3	De gauche à droite	/	Oui	Division arithmétique	Arithmétique
3	De gauche à droite	%	Oui	Modulo (restant)	Arithmétique
4	De gauche à droite	+	Oui	Ajout arithmétique ou concaténation de chaîne	Arithmétique, chaîne
4	De gauche à droite	-	Oui	Soustraction arithmétique	Arithmétique
5	De gauche à droite	<<	Oui	Décalage gauche	Binaire
5	De gauche à droite	>>	Oui	Décalage droite	Binaire
6	De gauche à droite	<	Oui	Inférieur à	Comparaison
6	De gauche à droite	<=	Oui	Inférieur ou égal à	Comparaison
6	De gauche à droite	>	Oui	Supérieur à	Comparaison
6	De gauche à droite	>=	Oui	Supérieur ou égal à	Comparaison
7	De gauche à droite	==	Oui	Egal à	Comparaison

Priorité	Associativité	Opérateur	Prise en charge en dehors des attributs METHOD	Description	Catégorie
7	De gauche à droite	!=	Oui	Pas égal à	Comparaison
8	De gauche à droite	&	Oui	Binaire et	Binaire
9	De gauche à droite	^	Oui	XOR (ou exclusif) binaire	Binaire
10	De gauche à droite		Oui	OU binaire	Binaire
11	De gauche à droite	&&	Oui	ET logique	Arithmétique
12	De gauche à droite		Oui	OU logique	Arithmétique
13	De droite à gauche	?:	Oui	Condition ternaire	Sous condition
13	De droite à gauche	=	Non	Affectation directe	Affectation
13	De droite à gauche	+=	Non	Affectation par somme	Affectation
13	De droite à gauche	-=	Non	Affectation par différence	Affectation
13	De droite à gauche	*=	Non	Affectation par produit	Affectation
13	De droite à gauche	/=	Non	Affectation par quotient	Affectation
13	De droite à gauche	%=	Non	Affectation par modulo (restant)	Affectation
13	De droite à gauche	<<=	Non	Affectation par décalage gauche binaire	Affectation
13	De droite à gauche	>>=	Non	Affectation par décalage droite binaire	Affectation
13	De droite à gauche	&=	Non	Affectation par et binaire	Affectation
13	De droite à gauche	^=	Non	Affectation par XOR binaire	Affectation
13	De droite à gauche	=	Non	Affectation par ou binaire	Affectation
14	De gauche à droite	,	Non	Séparation d'expressions	Enoncé

Le Tableau A.5 définit les opérations qui peuvent être réalisées pour les attributs VARIABLE ou les variables locales METHOD en fonction de leurs types de données.

Tableau A.5 – Opérations pour les attributs VARIABLE ou les variables locales METHOD

Type de VARIABLE	Type de données locales METHOD	Opérateurs
BOOLEAN	N.a.	== != && ! =
DOUBLE, FLOAT	double, float	+ - * / < <= == != >= > = += -= *= /=
INTEGER, UNSIGNED_INTEGER	char, int, long, long long, unsigned char, unsigned int, unsigned long, unsigned long long	+ - * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
DATE	N.a.	== != < <= >= > =
DATE_AND_TIME	N.a.	== != < <= >= > =
DURATION	N.a.	== != < <= >= > =
TIME	N.a.	== != < <= >= > =
TIME_VALUE(8)	N.a.	== != < <= >= > =
TIME_VALUE(4)	N.a.	== != < <= >= > =
BIT_ENUMERATED	unsigned char, unsigned int, unsigned long, unsigned long long	+ - * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
ENUMERATED	unsigned char, unsigned int, unsigned long, unsigned long long	+ - * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
INDEX	unsigned char, unsigned int, unsigned long, unsigned long long	+ - * / % < <= == != >= > << >> & ^ ~ = += -= *= /= &= = ^= ~= %= &= <<= >>=
OBJECT_REFERENCE	N.a.	== != = ->

Type de VARIABLE	Type de données locales METHOD	Opérateurs
ASCII, PACKED_ASCII, PASSWORD, VISIBLE	DD_STRING char []	+ == != [] =
BITSTRING	unsigned char[]	== !=
EUC	DD_STRING char []	+ == != [] =
OCTET	unsigned char[]	+ == != [] =
N.a.	time_t	== != =
Légende: N.a.: non applicable		

La concaténation des littéraux de chaîne sans opérateur + doit aussi être prise en charge (voir A.6.36, Attributs communs "string_literal").

A.4 Mots clés

Le Tableau A.6 contient les mots clés EDDL.

Tableau A.6 – Mots clés EDDL

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
ACCESS	ACTUATOR	ACTUATOR_ELECTRIC
ACTUATOR_ELECTRO_PNEUMATIC	ACTUATOR_HYDRAULIC	ADD
ADDRESSING	ALARM	ALIGN_LEFT
ALIGN_RIGHT	ANALOG_INPUT	ANALOG_OUTPUT
AO	AO1	AO2
AO3	AO4	AO5
API	ARRAY	ARRAY_INDEX
ARRAYS	ASCII	AUTO_CREATE
AXES	AXIS	AXIS_ITEMS
BAD	BIT_ENUMERATED	BITSTRING
BLOB	BLOBS	BLOCK
BLOCKS	BOOLEAN	break
BYTE_ORDER		
CAN_DELETE	CAPACITY	case
CASE	CATALOG	char
CHARACTERISTICS	CHART	CHART_ITEMS
CHARTS	CHECK_CONFIGURATION	CHILD

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
CHILD_COMPONENT	CLASS	CLASSIFICATION
COLLECTION	COLLECTION_ITEMS	COLLECTIONS
COLUMNBREAK	COMM_ERROR	COMMAND
COMMANDS	COMMANDS	COMPONENT
COMPONENT_FOLDER	COMPONENT_FOLDERS	COMPONENT_PARENT
COMPONENT_PATH	COMPONENT_REDEFINITIONS	COMPONENT_REFERENCE
COMPONENT_REFERENCES	COMPONENT_RELATION	COMPONENT_RELATIONS
COMPONENTS	COMPUTATION	CONNECTION_POINT
CONSTANT_UNIT	CONTAINED	continue
CONTROLLER	CONVERTER	CORRECTABLE
CORRECTION	COUNT	CYCLE_TIME
DATA	DATA_ENTRY_ERROR	DATA_ENTRY_WARNING
DATA1	DATA2	DATA3
DATA4	DATA5	DATA6
DATA7	DATA8	DATA9
DATE	DATE_AND_TIME	DD_ITEM
DD_REVISION	DD_STRING	DECLARATION
default	DEFAULT	DEFAULT_VALUE
DEFAULT_VALUES	DEFINITION	DELETE
DETAIL	DETECT	DEVICE
DEVICE_REVISION		DEVICE_TYPE
DIAGNOSE	DIAGNOSTIC	DIALOG
DIGITAL_INPUT	DIGITAL_OUTPUT	DISCRETE
DISCRETE_IN	DISCRETE_INPUT	DISCRETE_OUT
DISCRETE_OUTPUT	DISPLAY_FORMAT	DISPLAY_ITEMS
double	DOUBLE	DURATION
DV	DV1	DV2
DV3	DV4	DYNAMIC
EDD	EDD_PROFILE	EDD_REVISION
EDD_VERSION	EDIT_DISPLAY	EDIT_DISPLAY_ITEMS
EDIT_DISplays	EDIT_FORMAT	EDIT_ITEMS
ELECTRICAL_DISTRIBUTION	ELECTRICAL_DISTRIBUTION_POWER_MONITORING	ELEMENTS
else	ELSE	EMPHASIS
ENUMERATED	ERROR	EUC
EVENT	EVERYTHING	EXIT_ACTIONS
FALSE	FF	FILE
FILE_ITEMS	FILES	FILTER
FIRST	float	FLOAT
for	FREQUENCY	FREQUENCY_CONVERTER
FREQUENCY_INPUT	FREQUENCY_OUTPUT	FUNCTION
GAUGE	GRAPH	GRAPH_ITEMS
GRAPHS	GRID	GRID_ITEMS
GRIDS	GROUP	

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
HANDLING	HARDWARE	HART
HEADER	HEIGHT	HELP
HIDDEN	HIGH_HIGH_LIMIT	HIGH_LIMIT
HORIZONTAL	HORIZONTAL_BAR	HOURS
IDENTITY	if	IF
IGNORE_IN_HANDHELD	IGNORE_IN_HOST	IGNORE_IN_TEMPORARY_MASTER
IMAGE	IMAGE_ITEMS	IMAGES
IMPORT	INDEX	INDICATOR
INFO	INIT_ACTIONS	INITIAL_VALUE
INITIAL_VALUES	INLINE	INPUT
int	INTEGER	INTERFACE
INTERFACES	IS_CONFIG	ITEM_ARRAY
ITEM_ARRAY_ITEMS	ITEM_ARRAYS	ITEMS
KEY_POINTS		
LABEL	LARGE	LAST
LENGTH	LIKE	LINE_COLOR
LINE_TYPE	LINEAR	LINK
LIST	LIST_ITEMS	LISTS
LOAD_TO_APPLICATION	LOAD_TO_DEVICE	LOCAL
LOCAL_DISPLAY	LOCAL_PARAM	LOCAL_PARAMETERS
LOGARITHMIC	long	LOW_LIMIT
LOW_LOW_LIMIT		
MANUFACTURER	MANUFACTURER_EXT	MARGINAL
MAX_VALUE	MAX_VALUE1	MAX_VALUE2
MAX_VALUE3	MAX_VALUE4	MAX_VALUE5
MAX_VALUE6	MAX_VALUE7	MAX_VALUE8
MAX_VALUE9	MIN_VALUE	MIN_VALUE1
MIN_VALUE2	MIN_VALUE3	MIN_VALUE4
MIN_VALUE5	MIN_VALUE6	MIN_VALUE7
MIN_VALUE8	MIN_VALUE9	MINIMUM_NUMBER
MINUTES	MISC	MISC_ERROR
MISC_WARNING	MODE	MODE_AUTOMATIC
MODE_CASCADE	MODE_ERROR	MODE_ERROR
MODE_INITIALIZATION	MODE_LOCAL_OVERRIDE	MODE_MANUAL
MODE_OUT_OF_SERVICE	MODE_REMOTE_CASCADE	MODE_REMOTE_OUTPUT
MORE		
NETWORK	NETWORK_COMMUNICATION_SERVICE_PROVIDER	NETWORK_COMPONENT
NETWORK_COMPONENT	NETWORK_COMPONENT_WIRELESSADAPTER	NETWORK_CONNECTION_POINT
NEXT	NEXT_COMPONENT	NEXT_COMPONENT
NO_LABEL	NO_UNIT	NUMBER
NUMBER_OF_ELEMENTS	NUMBER_OF_POINTS	
OBJECT_REFERENCE	OCTET	OF
OFFLINE	ON_UPDATE_ACTIONS	ONLINE

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
OPERATE	OPERATION	OPTIONAL
ORIENTATION	OUTPUT	
PACKED_ASCII	PAGE	PARAM
PARAM_LIST	PARAMETER_LISTS	PARAMETERS
PARENT	PARENT_COMPONENT	PASSWORD
PATH	PHYSICAL	PI
PLUGIN	PLUGIN_ITEMS	PLUGINS
POST_EDIT_ACTIONS	POST_READ_ACTIONS	POST_READ_RECEIVE_ACTIONS
POST_READ_UPDATE_ACTIONS	POST_USERCHANGE_ACTIONS	POST_WRITE_ACTIONS
PRE_EDIT_ACTIONS	PRE_READ_ACTIONS	PRE_WRITE_ACTIONS
PREV	PREV_COMPONENT	PRIVATE
PROCESS	PROCESS_ERROR	PROCESS_VALUE
PROFIBUS_DP	PROFIBUS_PA	PROFINET
PROTOCOL		
READ	READ_ONLY	RECORD
RECORDER	RECORDS	REDEFINE
REDEFINITIONS	REDUNDANCY	REFRESH
REFRESH_ACTIONS	REFRESH_ITEMS	REFRESHES
RELATION_TYPE	RELATIONS	REMOTEIO
REPLY	REQUEST	REQUIRED_INTERFACE
REQUIRED_RANGES	RESPONSE_CODES	return
REVIEW	ROWBREAK	
SCALING	SCALING_FACTOR	SCAN
SCAN_LIST	SCOPE	SECONDS
SELECT	SELF	SELF_CORRECTING
SEMICONDUCTOR	SENSOR	SENSOR_ACOUSTIC
SENSOR_ANALYTIC	SENSOR_ANALYTIC_GAS	SENSOR_ANALYTIC_LIQUID
SENSOR_CONCENTRATION	SENSOR_DENSITY	SENSOR_FLOW
SENSOR_FLOW_CORIOLIS	SENSOR_FLOW_ELECTRO_MAGNETIC	SENSOR_FLOW_MECHANICAL
SENSOR_FLOW_RADIOMETRIC	SENSOR_FLOW_THERMAL	SENSOR_FLOW_ULTRASONIC
SENSOR_FLOW_VORTEX_COUNTER	SENSOR_LEVEL	SENSOR_LEVEL_BUOYANCY
SENSOR_LEVEL_CAPACITIVE	SENSOR_LEVEL_ECHO	SENSOR_LEVEL_HYDROSTATIC
SENSOR_LEVEL_RADIOMETRIC	SENSOR_MULTIVARIABLE	SENSOR_POSITION
SENSOR_PRESSURE	SENSOR_TEMPERATURE	SEPARATOR
SERVICE	SHARED	short
SIBLING_COMPONENT	signed	SIMULATION
SLOT	SMALL	SOFTWARE
SOURCE	SOURCE_ITEMS	SOURCES
SPECIALIST	STATE	STRIP
STYLE	SUB_SLOT	SUCCESS
SUMMARY	SUPPLIED_INTERFACE	SWEEP
switch	SWITCHGEAR	
TABLE	TEMPLATE	TEMPLATES

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
TEMPORARY	TIME	TIME_FORMAT
TIME_SCALE	time_t	TIME_VALUE
TRANSACTION	TRANSDUCER	TRANSPARENT
TRUE	TUNE	TV
TV1	TV2	TV3
TV4	TV5	TV6
TV7	TV8	TV9
TV10	TV11	TV12
TV13	TV14	TV15
TV16	TV17	TV18
TV19	TV20	TYPE
UNCORRECTABLE	UNIT	UNIT_ITEMS
UNITS	UNIVERSAL	unsigned
UNSIGNED_INTEGER	UUID	
VALIDITY	VARIABLE	VARIABLE_LIST
VARIABLE_LISTS	VARIABLE_STATUS	VARIABLES
VECTORS	VERTICAL	VERTICAL_BAR
VIEW_MAX	VIEW_MIN	VISIBILITY
VISIBLE		
WARNING	WAVEFORM	WAVEFORM_ITEMS
WAVEFORMS	while	WIDTH
WINDOW	WRITE	WRITE_AS_ONE
WRITE_AS_ONE_ITEMS	WRITE_AS_ONES	
X_AXIS	X_INCREMENT	X_INITIAL
X_LARGE	X_SMALL	X_VALUES
XX_LARGE	XX_SMALL	XXX_SMALL
XY		
Y_AXIS	Y_VALUES	YT

A.5 Terminaux

Le texte suivant contient les terminaux admis de la structure lexicale EDDL.

```

DEFINE digit          = { 0-9 } .
    bin_digit         = { 0 1 } .
    non_zero_digit    = { 1-9 '-' } .
    oct_digit         = { 0-7 } .
    hex_digit         = { 0-9abcdefABCDEF } .
    letter            = { a-zA-Z } .
    escapes           = { '\"?afnrtv'\" } .
    ISOLatin1char     = - { " } .

/* Integer */
(0b|0B)    bin_digit +      /* binary */
non_zero_digit digit *     /* decimal */
"0" oct_digit *             /* octal */
(0x|0X) hex_digit +        /* hexadecimal */

```