

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Devices and integration in enterprise systems – Function blocks (FB) for
process control and electronic device description language (EDDL) –
Part 3: EDDL syntax and semantics**

**Les dispositifs et leur intégration dans les systèmes de l'entreprise – Blocs
fonctionnels (FB) pour les procédés industriels et le langage de description
électronique de produit (EDDL) –
Partie 3: Sémantique et syntaxe EDDL**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2020 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
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 corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform

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 It also gives information on projects, replaced and withdrawn publications.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 once a month by email.

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: sales@iec.ch.

Electropedia - www.electropedia.org

The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

67 000 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.

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.

Recherche de publications IEC -

webstore.iec.ch/advsearchform

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 une fois par mois par email.

Service Clients - webstore.iec.ch/csc

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

Electropedia - www.electropedia.org

Le premier dictionnaire d'électrotechnologie en ligne au monde, avec plus de 22 000 articles terminologiques en anglais et en français, ainsi que les termes équivalents dans 16 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

67 000 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.

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Devices and integration in enterprise systems – Function blocks (FB) for
process control and electronic device description language (EDDL) –
Part 3: EDDL syntax and semantics**

**Les dispositifs et leur intégration dans les systèmes de l'entreprise – Blocs
fonctionnels (FB) pour les procédés industriels et le langage de description
électronique de produit (EDDL) –
Partie 3: Sémantique et syntaxe EDDL**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 25.040.40; 35.240.50

ISBN 978-2-8322-8468-1

**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.....	21
INTRODUCTION.....	24
1 Scope.....	25
2 Normative references	25
3 Terms, definitions, abbreviated terms and acronyms	26
3.1 Terms and definitions.....	26
3.2 Abbreviated terms and acronyms	28
4 Conformance statement.....	29
5 EDD and EDDL model	29
5.1 Overview of EDD and EDDL.....	29
5.2 EDD architecture	29
5.3 Concepts of EDD	29
5.4 Principles of the EDD development process.....	30
5.4.1 General	30
5.4.2 EDD source generation.....	30
5.4.3 EDD preprocessing.....	31
5.4.4 EDD compilation.....	31
5.5 Interrelations between the lexical structure and formal definitions.....	31
5.6 Builtins.....	31
5.7 Profiles	31
6 Conventions	31
6.1 General.....	31
6.2 Conventions for basic construct description	32
6.3 Conventions for lexical structure	32
7 Overview	33
7.1 EDDL features	33
7.2 Syntax representation	34
7.3 EDD language elements	34
7.4 Basic construction elements.....	34
7.4.1 General	34
7.4.2 AXIS.....	34
7.4.3 BLOB.....	34
7.4.4 BLOCK_A.....	35
7.4.5 BLOCK_B.....	35
7.4.6 CHART	35
7.4.7 COLLECTION.....	35
7.4.8 COMMAND.....	36
7.4.9 COMPONENT.....	36
7.4.10 COMPONENT_FOLDER.....	37
7.4.11 COMPONENT_REFERENCE.....	37
7.4.12 COMPONENT_RELATION.....	37
7.4.13 EDIT_DISPLAY	38
7.4.14 FILE	38
7.4.15 GRAPH.....	38
7.4.16 GRID	39
7.4.17 IMAGE.....	39

7.4.18	IMPORT	39
7.4.19	INTERFACE	39
7.4.20	LIKE	39
7.4.21	LIST	40
7.4.22	MENU	40
7.4.23	METHOD	41
7.4.24	PLUGIN	41
7.4.25	RECORD	41
7.4.26	REFERENCE_ARRAY	42
7.4.27	Relations	42
7.4.28	RESPONSE_CODES	43
7.4.29	SOURCE	43
7.4.30	TEMPLATE	44
7.4.31	VALUE_ARRAY	44
7.4.32	VARIABLE	44
7.4.33	VARIABLE_LIST	45
7.4.34	WAVEFORM	45
7.5	Common attributes	46
7.6	Special elements	46
7.7	Rules for instances	46
7.8	Rules for a list of VARIABLES	46
8	EDD identification information	47
8.1	General structure	47
8.2	DD_REVISION	48
8.3	DEVICE_REVISION	48
8.4	DEVICE_TYPE	49
8.5	EDD_PROFILE	49
8.6	EDD_VERSION	50
8.7	LAYOUT_TYPE	50
8.8	MANUFACTURER	50
8.9	MANUFACTURER_EXT	51
9	Data modelling	51
9.1	VARIABLE	51
9.1.1	General structure	51
9.1.2	CLASS	53
9.1.3	CONSTANT_UNIT	56
9.1.4	DEFAULT_VALUE	57
9.1.5	INITIAL_VALUE	57
9.1.6	POST_EDIT_ACTIONS	58
9.1.7	POST_READ_ACTIONS	58
9.1.8	POST_RQSTUPDATE_ACTIONS	59
9.1.9	POST_USERCHANGE_ACTIONS	59
9.1.10	POST_WRITE_ACTIONS	60
9.1.11	PRE_EDIT_ACTIONS	60
9.1.12	PRE_READ_ACTIONS	60
9.1.13	PRE_WRITE_ACTIONS	61
9.1.14	REFRESH_ACTIONS	61
9.1.15	TYPE	62
9.2	REFERENCE_ARRAY	81

9.2.1	General structure.....	81
9.2.2	ELEMENTS	84
9.2.3	item-type	84
9.3	VALUE_ARRAY	84
9.3.1	General structure.....	84
9.3.2	NUMBER_OF_ELEMENTS	85
9.3.3	TYPE	86
9.4	LIST	87
9.4.1	General structure.....	87
9.4.2	CAPACITY	88
9.4.3	COUNT	88
9.4.4	TYPE	89
9.5	COLLECTION	90
9.5.1	General structure.....	90
9.5.2	item-type	91
9.6	FILE	92
9.6.1	General structure.....	92
9.6.2	IDENTITY	93
9.6.3	ON_UPDATE_ACTIONS	94
9.6.4	SHARED	94
9.7	Relations	95
9.7.1	REFRESH	95
9.7.2	SEMANTIC_MAP	96
9.7.3	UNIT	98
9.7.4	WRITE_AS_ONE	99
9.8	BLOB	100
9.8.1	General structure.....	100
9.8.2	IDENTITY	101
10	Communication	102
10.1	COMMAND	102
10.1.1	General structure.....	102
10.1.2	Command addressing	103
10.1.3	API	104
10.1.4	BLOCK_B	104
10.1.5	HEADER	105
10.1.6	INDEX	105
10.1.7	NUMBER	105
10.1.8	OPERATION	106
10.1.9	SLOT	107
10.1.10	SUB_SLOT	107
10.1.11	TRANSACTION	108
10.2	RECORD	114
10.3	RESPONSE_CODES	115
10.4	VARIABLE_LIST	116
11	User interface	117
11.1	MENU	117
11.1.1	General structure.....	117
11.1.2	ACCESS	118
11.1.3	EXIT_ACTIONS	119

11.1.4	INIT_ACTIONS	119
11.1.5	ITEMS	119
11.1.6	POST_EDIT_ACTIONS	121
11.1.7	POST_READ_ACTIONS	122
11.1.8	POST_WRITE_ACTIONS	122
11.1.9	PRE_EDIT_ACTIONS	123
11.1.10	PRE_READ_ACTIONS	123
11.1.11	PRE_WRITE_ACTIONS	123
11.1.12	STYLE	124
11.1.13	Sequence diagrams for actions	125
11.2	EDIT_DISPLAY	127
11.2.1	General structure	127
11.2.2	EDIT_ITEMS	127
11.2.3	DISPLAY_ITEMS	128
11.2.4	POST_EDIT_ACTIONS	128
11.2.5	PRE_EDIT_ACTIONS	129
11.3	IMAGE	129
11.3.1	General structure	129
11.3.2	LINK	131
11.3.3	PATH	131
11.4	GRID	132
11.4.1	General structure	132
11.4.2	ORIENTATION	134
11.4.3	VECTORS	135
11.5	CHART	136
11.5.1	General structure	136
11.5.2	CYCLE_TIME	143
11.5.3	LENGTH	144
11.5.4	TYPE	144
11.5.5	Sequence diagram for actions	146
11.5.6	CHART legend/help rules	147
11.6	SOURCE	149
11.6.1	General structure	149
11.6.2	EXIT_ACTIONS	150
11.6.3	INIT_ACTIONS	151
11.6.4	REFRESH_ACTIONS	151
11.6.5	Y_AXIS	152
11.7	AXIS	152
11.7.1	General structure	152
11.7.2	MIN_VALUE, MAX_VALUE	153
11.7.3	SCALING	154
11.7.4	CONSTANT_UNIT	154
11.8	GRAPH	155
11.8.1	General structure	155
11.8.2	CYCLE_TIME	163
11.8.3	X_AXIS	164
11.8.4	Sequence diagram for actions	164
11.9	WAVEFORM	165
11.9.1	General structure	165

11.9.2	EXIT_ACTIONS	167
11.9.3	INIT_ACTIONS	167
11.9.4	KEY_POINTS	168
11.9.5	REFRESH_ACTIONS	169
11.9.6	TYPE	169
11.9.7	Y_AXIS	172
11.10	PLUGIN	173
11.10.1	General structure	173
11.10.2	UUID	173
12	METHODs	174
12.1	General structure	174
12.2	ACCESS	175
12.3	CLASS	176
12.4	TYPE	177
13	Device modelling	178
13.1	BLOCK_A	178
13.1.1	General structure	178
13.1.2	AXIS_ITEMS	180
13.1.3	CHARACTERISTICS	181
13.1.4	CHARTS	181
13.1.5	CHART_ITEMS	182
13.1.6	COLLECTION_ITEMS	182
13.1.7	EDIT_DISPLAY_ITEMS	182
13.1.8	FILES	183
13.1.9	FILE_ITEMS	183
13.1.10	GRAPHS	183
13.1.11	GRAPH_ITEMS	184
13.1.12	GRIDS	184
13.1.13	GRID_ITEMS	184
13.1.14	IMAGE_ITEMS	185
13.1.15	ITEM_ARRAY_ITEMS	185
13.1.16	LISTS	185
13.1.17	LIST_ITEMS	186
13.1.18	LOCAL_PARAMETERS	186
13.1.19	MENUS	186
13.1.20	MENU_ITEMS	187
13.1.21	METHODS	187
13.1.22	METHOD_ITEMS	188
13.1.23	PARAMETERS	188
13.1.24	PARAMETER_LISTS	188
13.1.25	PLUGINS	189
13.1.26	PLUGIN_ITEMS	189
13.1.27	REFRESH_ITEMS	190
13.1.28	SEMANTIC_MAP_ITEMS	190
13.1.29	SOURCE_ITEMS	190
13.1.30	UNIT_ITEMS	191
13.1.31	WAVEFORM_ITEMS	191
13.1.32	WRITE_AS_ONE_ITEMS	191
13.2	BLOCK_B	192

13.2.1	General structure	192
13.2.2	NUMBER	192
13.2.3	TYPE	193
13.3	LIKE	193
13.4	IMPORT	194
13.5	TEMPLATE	197
13.5.1	General structure	197
13.5.2	DEFAULT_VALUES	198
13.6	COMPONENT	199
13.6.1	General structure	199
13.6.2	BYTE_ORDER	200
13.6.3	CAN_DELETE	201
13.6.4	CHECK_CONFIGURATION	201
13.6.5	COMPONENT_RELATIONS	201
13.6.6	CONNECTION_POINT	202
13.6.7	DETECT	202
13.6.8	EDD	202
13.6.9	INITIAL_VALUES	203
13.6.10	REDUNDANCY	203
13.6.11	SCAN	204
13.6.12	SCAN_LIST	204
13.6.13	PRODUCT_URI	204
13.7	COMPONENT_FOLDER	205
13.8	COMPONENT_REFERENCE	205
13.9	COMPONENT_RELATION	206
13.9.1	General structure	206
13.9.2	COMPONENTS	207
13.9.3	RELATION_TYPE	208
13.9.4	ADDRESSING	209
13.9.5	MAXIMUM_NUMBER	210
13.9.6	MINIMUM_NUMBER	210
13.9.7	REQUIRED_INTERFACE	210
13.10	INTERFACE	211
13.10.1	General structure	211
13.10.2	DECLARATION	211
14	Common attributes	212
14.1	CLASSIFICATION	212
14.2	COMPONENT_PARENT	214
14.3	COMPONENT_PATH	214
14.4	DEFINITION	214
14.5	EMPHASIS	215
14.6	HANDLING	216
14.7	HEIGHT	216
14.8	HELP	217
14.9	LABEL	218
14.10	LINE_COLOR	218
14.11	LINE_TYPE	219
14.12	MEMBERS	220
14.13	PRIVATE	221

14.14	PROTOCOL	221
14.15	RESPONSE_CODES	222
14.16	SUPPLIED_INTERFACE	223
14.17	VALIDITY	223
14.18	VISIBILITY	223
14.19	WIDTH	224
14.20	WRITE_MODE	224
15	Redefinitions	225
15.1	Overview	225
15.2	AXIS	227
15.3	BLOB	228
15.4	BLOCK_A	228
15.5	BLOCK_B	230
15.6	CHART	230
15.7	COLLECTION	230
15.8	COMMAND	231
15.9	COMPONENT	231
15.10	COMPONENT_FOLDER	232
15.11	COMPONENT_REFERENCE	233
15.12	COMPONENT_RELATION	233
15.13	EDIT_DISPLAY	234
15.14	FILE	234
15.15	GRAPH	235
15.16	GRID	235
15.17	IMAGE	236
15.18	INTERFACE	236
15.19	LIST	237
15.20	MENU	237
15.21	METHOD	238
15.22	PLUGIN	239
15.23	RECORD	239
15.24	REFERENCE_ARRAY	239
15.25	RESPONSE_CODES	240
15.26	SOURCE	240
15.27	TEMPLATE	241
15.28	VALUE_ARRAY	241
15.29	VARIABLE	242
15.30	VARIABLE_LIST	243
15.31	WAVEFORM	244
16	Referencing	245
16.1	Referencing an EDD instance	245
16.2	Referencing AXIS of a GRAPH, SOURCE, WAVEFORM	245
16.3	Referencing bits of a BIT_ENUMERATED VARIABLE	245
16.4	Referencing members of a CHART	246
16.5	Referencing members of a COLLECTION	246
16.6	Referencing COMPONENT instances	246
16.7	Referencing COMPONENT types	247
16.8	Referencing members of a FILE	247
16.9	Referencing members of a GRAPH	248

16.10	Referencing elements of a LIST	248
16.11	Referencing members of a RECORD	248
16.12	Referencing elements of a REFERENCE_ARRAY	249
16.13	Referencing members of a SOURCE	249
16.14	Referencing elements of a VALUE_ARRAY	249
16.15	Referencing members of a VARIABLE_LISTS	250
16.16	Referencing BLOCK_A CHARACTERISTICS	250
16.17	Referencing elements of BLOCK_A LOCAL_PARAMETERS	250
16.18	Referencing elements of BLOCK_A PARAMETERS	251
16.19	Referencing elements of BLOCK_A PARAMETER_LISTS	251
16.20	Referencing CHARACTERISTICS of specific BLOCK_A instance	251
16.21	Referencing CHARTS of specific BLOCK_A instance	252
16.22	Referencing FILES of specific BLOCK_A instance	252
16.23	Referencing GRAPHS of specific BLOCK_A instance	253
16.24	Referencing GRIDS of specific BLOCK_A instance	253
16.25	Referencing LISTS of specific BLOCK_A instance	253
16.26	Referencing LOCAL_PARAMETERS of specific BLOCK_A instance	254
16.27	Referencing MENUS of specific BLOCK_A instance	254
16.28	Referencing METHODS of specific BLOCK_A instance	255
16.29	Referencing PARAMETERS of specific BLOCK_A instance	255
16.30	Referencing PLUGINS of specific BLOCK_A instance	256
17	Strings	256
17.1	Specifying a string as a string literal	256
17.2	Specifying a string as a string variable	257
17.3	Specifying a string as an enumeration value	257
17.4	Specifying a string as a dictionary reference	257
17.5	Referencing HELP and LABEL attributes of EDD instances	258
17.6	String operations	258
18	Conditional expression	259
18.1	General structure	259
18.2	IF conditional	259
18.3	SELECT conditional	260
19	Expression	260
19.1	General structure	260
19.2	Primary expressions	261
19.3	Unary expressions	263
19.4	Binary expressions	264
19.4.1	General structure	264
19.4.2	Multiplicative operators	264
19.4.3	Additive operators	264
19.4.4	Shift operators	265
19.4.5	Relational operators	265
19.4.6	Equality operators	265
19.4.7	Bitwise AND operator (&)	265
19.4.8	Bitwise XOR operator (^)	266
19.4.9	Bitwise OR operator ()	266
19.4.10	Logical AND operator (&&)	266
19.4.11	Logical OR operator ()	266
19.4.12	Conditional evaluation	266

20	Text dictionary	266
Annex A	(normative) EDDL formal definition	268
A.1	EDDL preprocessor	268
A.1.1	General structure	268
A.1.2	Directives	268
A.1.3	Predefined macros	271
A.1.4	NEWLINE characters	272
A.1.5	Comments	272
A.2	Conventions	272
A.2.1	Integer constants	272
A.2.2	Floating-point constants	272
A.2.3	String literals	273
A.2.4	Using language and country codes in string literals	273
A.3	Operators	274
A.4	Keywords	278
A.5	Symbols and symbol scope	282
A.6	Terminals	282
A.7	Formal EDDL syntax	283
A.7.1	General	283
A.7.2	EDD identification information	283
A.7.3	AXIS	285
A.7.4	BLOCK_A and BLOCK_B	286
A.7.5	CHART	290
A.7.6	COLLECTION	291
A.7.7	COMMAND	292
A.7.8	COMPONENT	295
A.7.9	COMPONENT_FOLDER	298
A.7.10	COMPONENT_REFERENCE	298
A.7.11	COMPONENT_RELATION	299
A.7.12	EDIT_DISPLAY	300
A.7.13	FILE	302
A.7.14	GRAPH	302
A.7.15	GRID	302
A.7.16	IMAGE	303
A.7.17	INTERFACE	304
A.7.18	LIST	304
A.7.19	IMPORT	305
A.7.20	LIKE	307
A.7.21	MENU	309
A.7.22	METHOD	310
A.7.23	RECORD	312
A.7.24	REFERENCE_ARRAY	312
A.7.25	Relations	313
A.7.26	RESPONSE_CODES	315
A.7.27	SOURCE	316
A.7.28	TEMPLATE	316
A.7.29	VALUE_ARRAY	316
A.7.30	VARIABLE	317
A.7.31	VARIABLE_LIST	328

A.7.32	WAVEFORM.....	328
A.7.33	Common attributes	330
A.7.34	Expression	334
A.7.35	C-Grammar.....	336
A.7.36	Redefinition	340
A.7.37	References	365
A.7.38	PLUGIN	367
A.7.39	BLOB.....	368
A.7.40	SEMANTIC_MAP.....	368
A.8	Formal dictionary syntax	369
Annex B (normative)	Data types.....	370
B.1	METHOD DEFINITION data types.....	370
B.2	VARIABLE TYPE data types	370
B.2.1	General	370
B.2.2	Coding of data DATE	372
B.2.3	Coding of data DATE_AND_TIME.....	373
B.2.4	Coding of data DURATION	373
B.2.5	Coding of data TIME	374
B.2.6	Coding of data TIME_VALUE.....	374
B.2.7	Coding of PACKED_ASCII (6-BIT ASCII) DATA FORMAT.....	375
B.2.8	Coding of BOOLEAN	376
B.2.9	Coding of BIT_ENUMERATED.....	376
Annex C (informative)	EDD example	378
C.1	EDD example of a temperature transmitter.....	378
C.2	EDD example.....	379
Bibliography		392
Figure 1	– Position of IEC 61804 in relation to other standards and products.....	24
Figure 2	– EDD generation process	30
Figure 3	– BLOCK_A	35
Figure 4	– CHART	35
Figure 5	– COLLECTION	36
Figure 6	– COMMAND	36
Figure 7	– COMPONENT	37
Figure 8	– COMPONENT_FOLDER	37
Figure 9	– COMPONENT_REFERENCE	37
Figure 10	– COMPONENT_RELATION	38
Figure 11	– EDIT_DISPLAY	38
Figure 12	– FILE.....	38
Figure 13	– GRAPH	39
Figure 14	– GRID.....	39
Figure 15	– IMAGE	39
Figure 16	– LIKE.....	40
Figure 17	– LIST.....	40
Figure 18	– MENU	41
Figure 19	– RECORD.....	41

Figure 20 – REFERENCE_ARRAY	42
Figure 21 – REFRESH	42
Figure 22 – SEMANTIC_MAP	43
Figure 23 – UNIT	43
Figure 24 – WRITE_AS_ONE.....	43
Figure 25 – SOURCE.....	44
Figure 26 – VALUE_ARRAY	44
Figure 27 – VARIABLE	45
Figure 28 – VARIABLE_LIST	45
Figure 29 – WAVEFORM	45
Figure 30 – REFERENCE_ARRAY example.....	83
Figure 31 – Abstract COMMAND definition example	83
Figure 32 – EDD examples of SEMANTIC_MAP	98
Figure 33 – Example of an item mask	111
Figure 34 – Example of a single item mask	111
Figure 35 – Mapping example with a single item mask	112
Figure 36 – Example of multiple item masks	112
Figure 37 – Mapping example with a multiple item mask	112
Figure 38 – INFO qualifier.....	113
Figure 39 – INDEX qualifier	113
Figure 40 – INFO and INDEX qualifier	114
Figure 41 – Examples of MENU STYLES.....	125
Figure 42 – MENU activation	126
Figure 43 – EDD example of an IMAGE	130
Figure 44 – EDD example of an IMAGE with the LINK attribute	130
Figure 45 – EDD example of a GRID.....	134
Figure 46 – Result of the EDD example	134
Figure 47 – Example of a chart with one curve in a dialog.....	138
Figure 48 – Example of a chart with two SOURCES	139
Figure 49 – Displaying example of a chart with two SOURCES.....	140
Figure 50 – Example of a chart with three horizontal bars	141
Figure 51 – Displaying example of a chart with three horizontal bars	141
Figure 52 – Example of a chart in a dialog	143
Figure 53 – Trend CHART types examples	146
Figure 54 – Meter CHART types examples.....	146
Figure 55 – CHART activation.....	147
Figure 56 – Legend/help for SOURCES with single and multiple VARIABLES – Y_AXIS not defined.....	148
Figure 57 – Legend/help for SOURCES arrangement of multiple SOURCES with shared Y_AXIS.....	149
Figure 58 – A graph and the visual elements	156
Figure 59 – Example of a graph	159
Figure 60 – Multiple used axes	160
Figure 61 – EDD with device-supported zooming and scrolling	163

Figure 62 – GRAPH activation	165
Figure 63 – EDDL import mechanisms	195
Figure 64 – Example of TEMPLATE usage	198
Figure 65 – REQUIRED_RANGES usage.....	208
Figure 66 – NEXT_COMPONENT usage.....	209
Figure C.1 – Example of an operator screen using EDD.....	378
Table 1 – Format for the basic construct table	32
Table 2 – Contents of the basic construct table.....	32
Table 3 – Attribute usage descriptions	32
Table 4 – Rules for lexical convention.....	33
Table 5 – EDD sources	33
Table 6 – EDD identification information attributes.....	47
Table 7 – DD_REVISION attribute	48
Table 8 – DEVICE_REVISION attribute.....	48
Table 9 – DEVICE_TYPE attribute	49
Table 10 – EDD_PROFILE attribute	49
Table 11 – EDD_VERSION attribute	50
Table 12 – LAYOUT_TYPE attribute	50
Table 13 – MANUFACTURER attribute	51
Table 14 – MANUFACTURER_EXT attribute.....	51
Table 15 – VARIABLE attributes	52
Table 16 – CLASS attributes.....	53
Table 17 – CONSTANT_UNIT attribute.....	56
Table 18 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS attributes	58
Table 19 – POST_USERCHANGE_ACTIONS, POST_RQSTUPDATE_ACTIONS attributes	59
Table 20 – TYPE overview.....	62
Table 21 – MIN_VALUE, MAX_VALUE attribute	63
Table 22 – TYPE attributes for DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER.....	65
Table 23 – Default display format conversion type	65
Table 24 – Default width/precision based on VARIABLE and format conversion type	66
Table 25 – TYPE attributes for DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE.....	69
Table 26 – TYPE attributes for BIT_ENUMERATED	71
Table 27 – size attribute	72
Table 28 – status–class attributes.....	72
Table 29 – ALL, AO, DV, TV attributes	74
Table 30 – TYPE attributes for ENUMERATED	75
Table 31 – TYPE attributes for INDEX	76
Table 32 – TYPE attributes for ASCII, EUC, PACKED_ASCII, PASSWORD, VISIBLE	78
Table 33 – String handling	78

Table 34 – TYPE attributes for OCTET	79
Table 35 – TYPE attributes for BITSTRING.....	81
Table 36 – REFERENCE_ARRAY attributes	82
Table 37 – ELEMENTS attribute	84
Table 38 – VALUE_ARRAY attributes	85
Table 39 – NUMBER_OF_ELEMENTS attribute	86
Table 40 – TYPE attribute.....	86
Table 41 – LIST attributes.....	87
Table 42 – CAPACITY attribute.....	88
Table 43 – COUNT attribute.....	89
Table 44 – TYPE attribute.....	90
Table 45 – COLLECTION attributes	91
Table 46 – item-type	91
Table 47 – FILE attributes.....	93
Table 48 – IDENTITY attribute	94
Table 49 – ON_UPDATE_ACTIONS attribute.....	94
Table 50 – SHARED attribute	94
Table 51 – REFRESH overview	95
Table 52 – REFRESH attributes.....	96
Table 53 – SEMANTIC_MAP overview	97
Table 54 – SEMANTIC_MAP attributes	97
Table 55 – UNIT overview.....	99
Table 56 – UNIT attributes.....	99
Table 57 – WRITE_AS_ONE overview.....	100
Table 58 – WRITE_AS_ONE attribute	100
Table 59 – BLOB attributes.....	101
Table 60 – IDENTITY attribute.....	101
Table 61 – COMMAND attributes	103
Table 62 – API attribute	104
Table 63 – BLOCK_B attribute	105
Table 64 – HEADER attribute	105
Table 65 – INDEX attribute	105
Table 66 – NUMBER attribute	106
Table 67 – OPERATION attribute.....	106
Table 68 – PROFIBUS and PROFINET communication mapping.....	107
Table 69 – SLOT attribute.....	107
Table 70 – SUB_SLOT attribute.....	107
Table 71 – TRANSACTION attributes.....	108
Table 72 – REPLY and REQUEST attributes.....	109
Table 73 – POST_RQSTRECEIVE_ACTIONS attributes	114
Table 74 – RECORD attributes	115
Table 75 – RESPONSE_CODES overview	116
Table 76 – RESPONSE_CODES attributes	116

Table 77 – VARIABLE_LIST attributes	117
Table 78 – MENU attributes	118
Table 79 – ACCESS attribute	118
Table 80 – MENU ITEMS	120
Table 81 – MENU ITEMS qualifiers	121
Table 82 – EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS attributes	122
Table 83 – STYLE attribute	124
Table 84 – EDIT_DISPLAY attributes	127
Table 85 – EDIT_ITEMS attribute	128
Table 86 – DISPLAY_ITEM attributes	128
Table 87 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS attribute	129
Table 88 – IMAGE attributes	130
Table 89 – LINK attribute	131
Table 90 – PATH attribute	131
Table 91 – Image formats	132
Table 92 – GRID attributes	132
Table 93 – ORIENTATION attribute	135
Table 94 – VECTORS attribute	135
Table 95 – CHART attributes	137
Table 96 – CYCLE_TIME attribute	144
Table 97 – LENGTH attribute	144
Table 98 – TYPE attribute	145
Table 99 – SOURCE attributes	150
Table 100 – Y_AXIS attribute	152
Table 101 – AXIS attributes	153
Table 102 – MIN_VALUE, MAX_VALUE attribute	154
Table 103 – SCALING attribute	154
Table 104 – GRAPH attributes	157
Table 105 – CYCLE_TIME attribute	164
Table 106 – X_AXIS attribute	164
Table 107 – WAVEFORM attributes	166
Table 108 – EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS attribute	167
Table 109 – KEY_POINTS attribute	168
Table 110 – X_VALUES, Y_VALUES attribute	169
Table 111 – TYPE attribute	170
Table 112 – XY attribute	170
Table 113 – YT attribute	171
Table 114 – HORIZONTAL attribute	172
Table 115 – VERTICAL attribute	172
Table 116 – Y_AXIS attribute	173
Table 117 – PLUGIN attributes	173
Table 118 – UUID attribute	174

Table 119 – METHOD attributes	175
Table 120 – Parameter types	175
Table 121 – ACCESS attributes	176
Table 122 – CLASS attributes	176
Table 123 – TYPE attributes	178
Table 124 – BLOCK_A attributes	180
Table 125 – AXIS_ITEMS attribute	181
Table 126 – CHARACTERISTICS attribute	181
Table 127 – CHARTS attribute	181
Table 128 – CHART_ITEMS attribute	182
Table 129 – COLLECTION_ITEMS attribute	182
Table 130 – EDIT_DISPLAY_ITEMS attribute	182
Table 131 – FILES attribute	183
Table 132 – FILE_ITEMS attribute	183
Table 133 – GRAPHS attribute	184
Table 134 – GRAPH_ITEMS attribute	184
Table 135 – GRIDS attribute	184
Table 136 – GRID_ITEMS attribute	185
Table 137 – IMAGE_ITEMS attribute	185
Table 138 – ITEM_ARRAY_ITEMS attribute	185
Table 139 – LISTS attribute	186
Table 140 – LIST_ITEMS attribute	186
Table 141 – MENUS attribute	187
Table 142 – MENU_ITEMS attribute	187
Table 143 – METHODS attribute	187
Table 144 – METHOD_ITEMS attribute	188
Table 145 – PARAMETERS attributes	188
Table 146 – PARAMETER_LISTS attributes	189
Table 147 – PLUGINS attribute	189
Table 148 – PLUGIN_ITEMS attribute	189
Table 149 – REFRESH_ITEMS attribute	190
Table 150 – SEMANTIC_MAP_ITEMS attribute	190
Table 151 – SOURCE_ITEMS attribute	190
Table 152 – UNIT_ITEMS attribute	191
Table 153 – WAVEFORM_ITEMS attribute	191
Table 154 – WRITE_AS_ONE_ITEMS attribute	191
Table 155 – BLOCK_B attributes	192
Table 156 – NUMBER attributes	193
Table 157 – TYPE attributes	193
Table 158 – LIKE overview	194
Table 159 – LIKE attributes	194
Table 160 – IMPORT overview	196
Table 161 – Importing device description	196

Table 162 – TEMPLATE attributes	198
Table 163 – DEFAULT_VALUES attribute	198
Table 164 – COMPONENT attributes	200
Table 165 – BYTE_ORDER attribute	200
Table 166 – CAN_DELETE attribute	201
Table 167 – CHECK_CONFIGURATION attribute	201
Table 168 – COMPONENT_RELATIONS attribute	202
Table 169 – CONNECTION_POINT attribute	202
Table 170 – DETECT attribute	202
Table 171 – EDD attribute	203
Table 172 – INITIAL_VALUES attribute	203
Table 173 – REDUNDANCY attribute	203
Table 174 – SCAN attribute	204
Table 175 – SCAN_LIST attribute	204
Table 176 – PRODUCT_URI attribute	204
Table 177 – COMPONENT_FOLDER attributes	205
Table 178 – COMPONENT_REFERENCE attributes	206
Table 179 – COMPONENT_RELATION attributes	207
Table 180 – COMPONENTS attribute	208
Table 181 – RELATION_TYPE attribute	209
Table 182 – ADDRESSING attribute	209
Table 183 – MAXIMUM_NUMBER attribute	210
Table 184 – MINIMUM_NUMBER attribute	210
Table 185 – REQUIRED_INTERFACE attribute	210
Table 186 – INTERFACE attributes	211
Table 187 – DECLARATION attribute	212
Table 188 – CLASSIFICATION attributes	212
Table 189 – COMPONENT_PARENT attribute	214
Table 190 – COMPONENT_PATH attribute	214
Table 191 – DEFINITION attribute	215
Table 192 – EMPHASIS attribute	216
Table 193 – HANDLING attribute	216
Table 194 – HEIGHT/WIDTH attribute	217
Table 195 – HELP attribute	218
Table 196 – LABEL attribute	218
Table 197 – LINE_COLOR attribute	219
Table 198 – Standard RGB color values	219
Table 199 – LINE_TYPE attribute	220
Table 200 – MEMBERS attributes	221
Table 201 – PRIVATE attributes	221
Table 202 – PROTOCOL attribute	222
Table 203 – RESPONSE_CODES attribute	222
Table 204 – SUPPLIED_INTERFACE attribute	223

Table 205 – VALIDITY attributes	223
Table 206 – VISIBILITY attributes	224
Table 207 – WRITE_MODE attributes	225
Table 208 – Redefinition attributes	227
Table 209 – Redefinition rules for AXIS attributes	228
Table 210 – Redefinition rules for BLOB attributes	228
Table 211 – Redefinition rules for BLOCK_A attributes	229
Table 212 – Redefinition rules for BLOCK_B attributes	230
Table 213 – Redefinition rules for CHART attributes	230
Table 214 – Redefinition rules for COLLECTION attributes	231
Table 215 – Redefinition rules for COMMAND attributes	231
Table 216 – Redefinition rules for COMPONENT attributes	232
Table 217 – Redefinition rules for COMPONENT_FOLDER attributes	232
Table 218 – Redefinition rules for COMPONENT_REFERENCE attributes	233
Table 219 – Redefinition rules for COMPONENT_RELATION attributes	234
Table 220 – Redefinition rules for EDIT_DISPLAY attributes	234
Table 221 – Redefinition rules for FILE attributes	235
Table 222 – Redefinition rules for GRAPH attributes	235
Table 223 – Redefinition rules for GRID attributes	236
Table 224 – Redefinition rules for IMAGE attributes	236
Table 225 – Redefinition rules for INTERFACE attributes	236
Table 226 – Redefinition rules for LIST attributes	237
Table 227 – Redefinition rules for MENU attributes	238
Table 228 – Redefinition rules for METHOD attributes	238
Table 229 – Redefinition rules for PLUGIN attributes	239
Table 230 – Redefinition rules for RECORD attributes	239
Table 231 – Redefinition rules for REFERENCE_ARRAY attributes	240
Table 232 – Redefinition rules for RESPONSE_CODES attributes	240
Table 233 – Redefinition rules for SOURCE attributes	241
Table 234 – Redefinition rules for TEMPLATE attributes	241
Table 235 – Redefinition rules for VALUE_ARRAY attributes	242
Table 236 – Redefinition rules for VARIABLE attributes	243
Table 237 – Redefinition rules for VARIABLE_LIST attributes	244
Table 238 – Redefinition rules for WAVEFORM attributes	244
Table 239 – Referencing an EDD instance	245
Table 240 – Referencing AXIS of a GRAPH. SOURCE, WAVEFORM	245
Table 241 – Referencing elements of VARIABLE	246
Table 242 – Referencing members of CHART	246
Table 243 – Referencing members of COLLECTION	246
Table 244 – Referencing a COMPONENT instance	247
Table 245 – Referencing a COMPONENT type	247
Table 246 – Referencing members of FILE	247
Table 247 – Referencing members of GRAPH	248

Table 248 – Referencing elements of LIST	248
Table 249 – Referencing elements of RECORD	249
Table 250 – Referencing members of REFERENCE_ARRAY	249
Table 251 – Referencing members of SOURCE	249
Table 252 – Referencing elements of VALUE_ARRAY	250
Table 253 – Referencing members of VARIABLE_LISTS	250
Table 254 – Referencing BLOCK_A CHARACTERISTICS	250
Table 255 – Referencing members of BLOCK_A LOCAL_PARAMETER.....	251
Table 256 – Referencing members of a BLOCK_A PARAMETERS.....	251
Table 257 – Referencing members of BLOCK_A PARAMETER_LISTS	251
Table 258 – Referencing CHARACTERISTICS of specific BLOCK_A instance	252
Table 259 – Referencing CHARTS of specific BLOCK_A instance	252
Table 260 – Referencing FILES of specific BLOCK_A instance	252
Table 261 – Referencing GRAPHS of specific BLOCK_A instance	253
Table 262 – Referencing GRIDS of specific BLOCK_A instance.....	253
Table 263 – Referencing LISTS of specific BLOCK_A instance.....	254
Table 264 – Referencing LOCAL_PARAMETERS of specific BLOCK_A instance	254
Table 265 – Referencing MENUS of specific BLOCK_A instance	255
Table 266 – Referencing METHODS of specific BLOCK_A instance	255
Table 267 – Referencing PARAMETERS of specific BLOCK_A instance	256
Table 268 – Referencing PLUGINS of specific BLOCK_A instance	256
Table 269 – String as a string literal.....	256
Table 270 – String as a string variable.....	257
Table 271 – String as an enumeration value	257
Table 272 – String as a dictionary reference	258
Table 273 – Referencing HELP and LABEL attributes of EDD instances	258
Table 274 – String operation.....	258
Table 275 – IF conditional.....	259
Table 276 – SELECT conditional	260
Table 277 – Primary expressions	261
Table 278 – Attribute values of VARIABLES.....	262
Table 279 – AXIS attribute values	263
Table 280 – BLOB attribute values.....	263
Table 281 – LIST attribute values	263
Table 282 – ARRAY attribute values	263
Table 283 – Unary expressions.....	264
Table 284 – Multiplicative operators.....	264
Table 285 – Additive operators	265
Table 286 – Shift operators.....	265
Table 287 – Relational operators	265
Table 288 – Equality operators	265
Table 289 – Text dictionary attributes	267
Table A.1 – Conventions for integer constants	272

Table A.2 – Using escape sequences in string literals.....	273
Table A.3 – Language code examples for string literals	274
Table A.4 – Precedence and associativity for EDDL operators	275
Table A.5 – Operations for VARIABLES or METHOD local variables	276
Table A.6 – EDDL keywords	278
Table B.1 – METHOD DEFINITION data types.....	370
Table B.2 – VARIABLE TYPES.....	371
Table B.3 – DATE coding.....	373
Table B.4 – DATE_AND_TIME coding.....	373
Table B.5 – DURATION coding	373
Table B.6 – DURATION coding for ISA100.....	374
Table B.7 – TIME coding.....	374
Table B.8 – TIME coding for ISA100	374
Table B.9 – TIME_VALUE coding (four octets).....	375
Table B.10 – TIME_VALUE coding (eight octets)	375
Table B.11 – PACKED_ASCII coding	376
Table B.12 – BOOLEAN coding	376
Table B.13 – BIT_ENUMERATED coding.....	377
Table B.14 – BIT_ENUMERATED coding for FF.....	377

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**DEVICES AND INTEGRATION IN ENTERPRISE SYSTEMS –
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 fourth edition cancels and replaces the third edition published in 2015. This edition constitutes a technical revision.

This edition was developed by merging material from multiple variants of existing EDDL specifications including those from FieldComm Group (FOUNDATION™ Fieldbus¹, HART®²), PROFIBUS™³ Nutzerorganisation e.V. (PNO), and ISA100_Wireless™⁴ Compliance Institute (ISA100 WCI). Any places where there may be a profile deviation are now indicated in the context where the related deviation is found. As a result, the formatting and numbering of this edition may be different from any of the individual specifications from which this edition was derived.

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

- Communication profiles ISA100 and GPE were added.
- EDD Identification Information has a new LAYOUT_TYPE attribute.
- New construct SEMANTIC_MAP was added.
- CLASS attribute values LOCAL_A and LOCAL_B were added.
- Extended LIST functionality to support device managed lists.

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

CDV	Report on voting
65E/631/CDV	65E/689/RVC

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 *Devices and integration in enterprise systems – Function blocks (FB) for process control and electronic device description language (EDDL)*, can be found on the IEC website.

Future parts of IEC 61804 will carry the new general title as cited above. Titles of existing parts will be updated at the time of the next edition.

¹ FOUNDATION™ Fieldbus is the trademark of FieldComm Group. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

² HART® is the registered trademark of FieldComm Group. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

³ PROFIBUS and PROFINET are the trademarks of the PROFIBUS Nutzerorganisation e.V. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

⁴ ISA100_Wireless™ is the trademark of ISA100 Wireless Compliance Institute. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

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.

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020

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 "Devices and integration in enterprise systems – 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 builtin library

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

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.

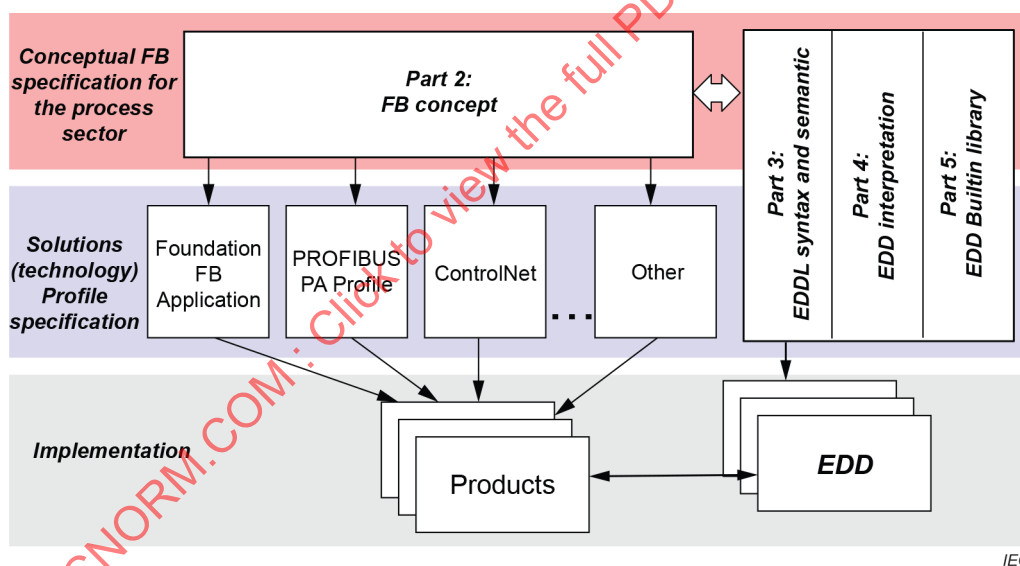


Figure 1 – Position of IEC 61804 in relation to other standards and products

DEVICES AND INTEGRATION IN ENTERPRISE SYSTEMS – 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 document 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 e.g. 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 document.

This document 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.

IEC 61804-4 specifies EDD interpretation for EDD applications and EDDs to support EDD interoperability.

IEC 61804-5 specifies the EDDL builtin library and provides the profiles of the various fieldbuses.

2 Normative references

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

IEC 60050-351, *International Electrotechnical Vocabulary (IEV) – Part 351: Control technology*

IEC 61804-2, *Function blocks (FB) for process control – Part 2: Specification of FB concept*

IEC 61804-4, *Function blocks (FB) for process control and electronic device description language (EDDL) – Part 4: EDD interpretation*

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 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 9899:1999, *Programming languages – C*

ISO/IEC 10646, *Information technology – Universal Coded Character Set (UCS)*

ISO/IEC 10918-1, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 15948, *Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification*

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*

ISO/TS 29002-5, *Industrial automation systems and integration – Exchange of characteristic data – Part 5: Identification scheme*

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

RFC 3629, *User Datagram Protocol*, available at <http://www.ietf.org/rfc/rfc0768.txt> [viewed 2018-12-21]

W3C Recommendation, *Cascading Style Sheets Level 2 Specification*, available at <http://www.w3.org/TR/CSS2> [viewed 2018-12-21]

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, IEC 61804-2 and the following apply.

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

- ISO Online browsing platform: available at <https://www.iso.org/obp>

- IEC Electropedia: available at <http://www.electropedia.org/>

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 61804-2:2018, 3.1.35]

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, modified – In the definition, "configurations" has been deleted.]

3.2 Abbreviated terms and acronyms

ASCII	American Standard Code for Information Interchange according to ISO/IEC 10646
CP	Communication Profile
CPF	Communication Profile Family
CS	Communication Server
EDD	Electronic Device Description
EDDL	Electronic Device Description Language
EUC	Extended Unicode according to ISO/IEC 10646
FB	Function Block
FF	FOUNDATION Fieldbus
GPE	Generic Protocol Extension
HART	Highway Addressable Remote Transducer
HMI	Human Machine Interface
I/O	Input/Output
ID	Identifier
ISA100	ISA100_Wireless
OSI	Open Systems Interconnection
PI	PROFIBUS and PROFINET International
PB	PROFIBUS
PDU	Protocol Data Unit

PN	PROFINET
PNO	PROFIBUS Nutzerorganisation
QM	Quality Management

4 Conformance statement

A statement of conformance to a profile of this document shall be stated as

– compliance to IEC 61804-3 <Type>

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 abbreviations used such as CS, FF, GPE, HART, ISA100, PB, or PN, 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 EDD and EDDL model

5.1 Overview of EDD and EDDL

An electronic device description (EDD) contains all device parameter(s) 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.

5.2 EDD architecture

From the viewpoint of the 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.

5.3 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;
- business logic (programmed 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 to match a specific EDD to a specific device. Manufacturer, device type, and revision are examples of information that can be used to match a device to a specific EDD. An example EDD is provided as a reference in Annex C.

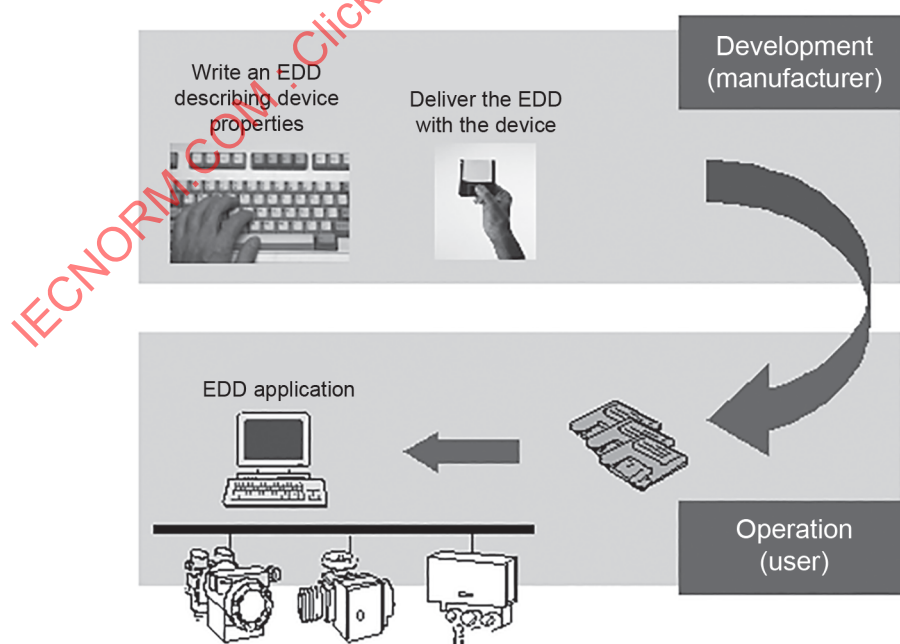
5.4 Principles of the EDD development process

5.4.1 General

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

5.4.2 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.

5.4.3 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.

5.4.4 EDD compilation

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

5.5 Interrelations between the lexical structure and formal definitions

The lexical structure of EDDL and its elements are described in Clauses 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20. 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 can be developed as an additional option.

5.6 Builtins

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

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

5.7 Profiles

EDDL is a unified specification of existing legacy EDD concepts. Concrete EDD applications use a subset of the EDDL specification. The elements of the specification applicable to various profiles are described in tables throughout this document, along with the description of the element.

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.

6 Conventions

6.1 General

In IEC 61804 (all parts), 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.

6.2 Conventions for basic construct description

The following conventions apply for the basic construct definition.

- Table 1 defines the format for the builtin overview table.
- Table 2 defines the contents of the columns of Table 1.
- Table 3 defines the usage of attributes of basic constructs or their subattributes.
- EDDL basic constructs and attributes referenced in this document are written in upper case letters.

Table 1 – Format for the basic construct table

Reference	Basic construct	Attribute	Communication profiles

Table 2 – Contents of the basic construct table

Column	Text	Meaning
Reference		Reference to the respective subclause that specifies the attribute of the basic construct.
Basic construct		Name of the basic construct.
Communication profiles	✓	The basic construct is supported by the respective communication profile.
	x	The basic construct is not supported by the respective communication profile.

Table 3 – Attribute usage 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 constrained and it is only present if the constraint is met.
L	Vvv	The presence of this attribute is limited to be used only in specified cases.

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 constrained. The constraint 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.

6.3 Conventions for lexical structure

The lexical structures in this document are specified using the rules defined in Table 4.

Table 4 – Rules for lexical convention

Lexical structure element	Description
bold	Concrete syntax literals are in bold .
<i>italics</i>	Syntax elements are in <i>italics</i> .
:=	Definition of a syntax element.
<cond>	The preceding syntax element may be conditionally specified.
[]	Grouping of syntax elements
*	The preceding syntax element may appear zero or more times. The syntax element can be a group of syntax elements enclosed in square brackets.
+	The preceding syntax element may appear one or more times. The syntax element can be a group of syntax elements enclosed in square brackets.
?	The preceding syntax element may appear zero or one time. The syntax element can be a group of syntax elements enclosed in square brackets.
	Alternative syntax elements
expression	An expression
<data type>-expression	An expression that can be resolved to a specified data type.
<data type>-constant	A constant value of the specified data type.
<data type>-value	A value of the specified data type.

Table 5 describes the applicable EDD sources for lexical structure element definitions.

Table 5 – EDD sources

Lexical structure element	Applicable source						
	Constant (Literal)	Referenced attribute	Enumerated()	Dictionary reference (only string)	Concatenation ('+', only string)	Expression incl. type converting	Condition (IF, SELECT)
<data type>-constant	x	x	x	x			
<data type>-value	x	x	x	x	x		
<data type>-value<cond>	x	x	x	x	x		x
<data type>-expression	x	x	x	x	x	x	
<data type>-expression<cond>	x	x	x	x	x	x	x

7 Overview

7.1 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). EDD developers are required to identify within each EDD details as to which profile has been used.

7.2 Syntax representation

The lexical structure 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 document, it is possible to use other syntax definitions, which may be added in the future to allow other features and representations for future progress.

7.3 EDD language elements

The language is structured with the following language elements:

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

Frequently used text strings, for example, help text and multilingual lists of labels should be separated in a text dictionary (see Clause 20).

Identification information is the first entry in every EDD file. 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.

7.4 Basic construction elements

7.4.1 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.

7.4.2 AXIS

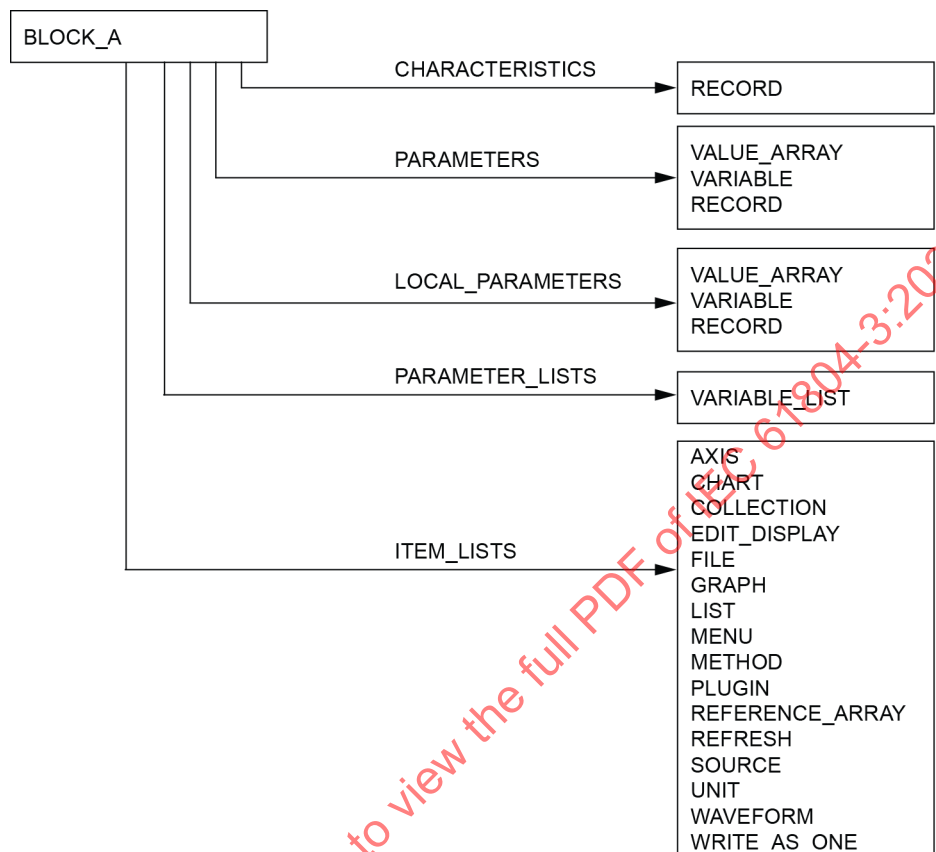
AXIS describes the axis of a CHART or GRAPH (see 11.7).

7.4.3 BLOB

BLOB describes a Binary Large Object used for transferring binary data to or from a device (see 9.8).

7.4.4 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 13.1 and Figure 3).



IEC

Figure 3 – BLOCK_A

7.4.5 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 13.2).

7.4.6 CHART

CHART describes a chart used to display data from a device in the EDD application (see 11.5 and Figure 4).



IEC

Figure 4 – CHART

7.4.7 COLLECTION

COLLECTION describes logical groupings of data (see 9.5 and Figure 5).

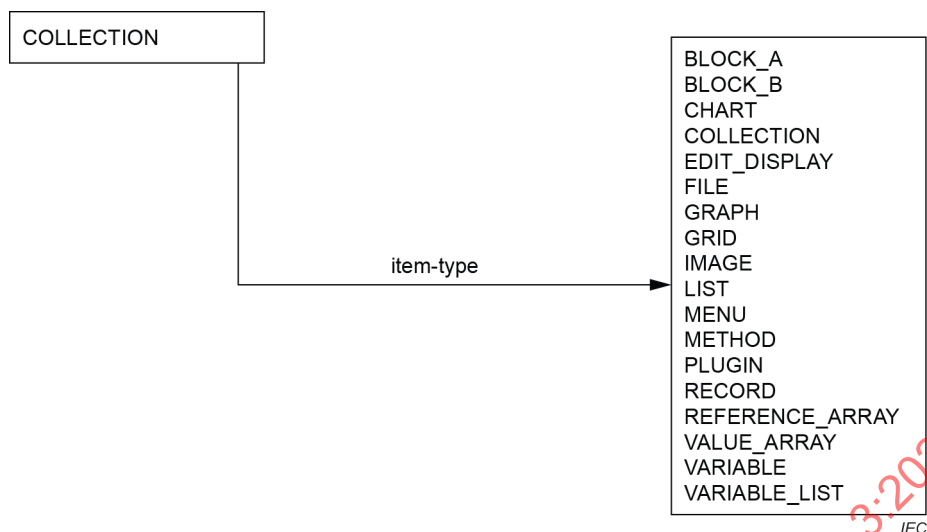


Figure 5 – COLLECTION

7.4.8 COMMAND

COMMAND describes the structure and the addressing of the variables in the device (see 10.1 and Figure 6).

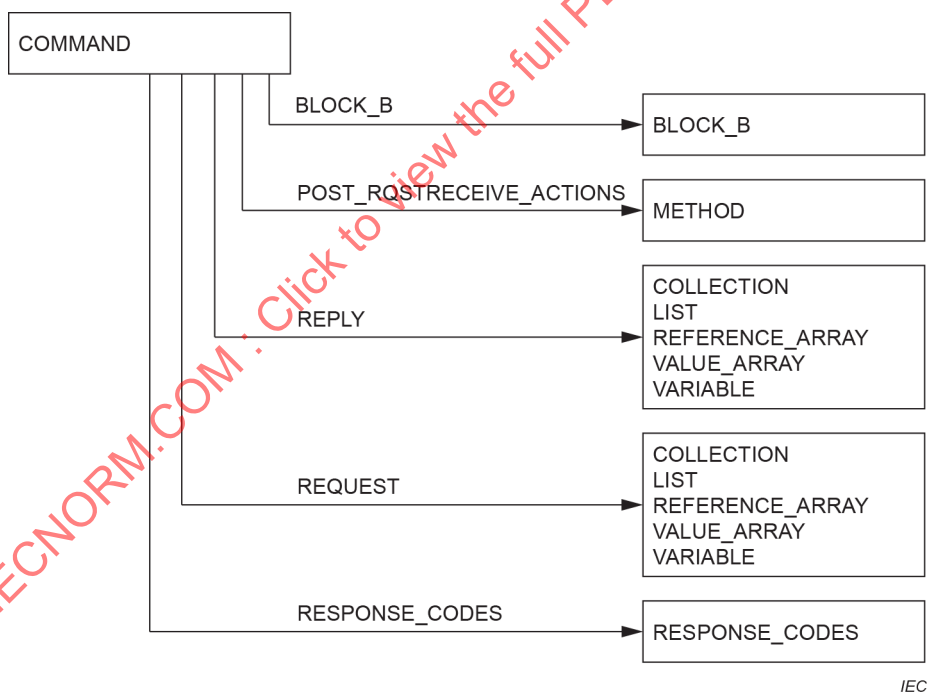
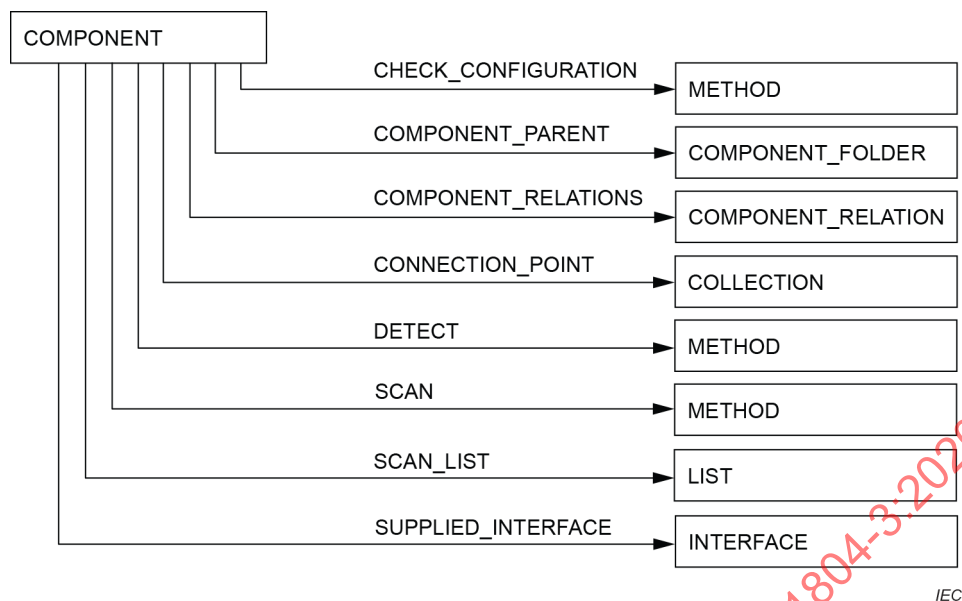


Figure 6 – COMMAND

7.4.9 COMPONENT

COMPONENT describes the configuration behavior of the device that is described in the EDD (see 13.6 and Figure 7).

**Figure 7 – COMPONENT****7.4.10 COMPONENT_FOLDER**

COMPONENT_FOLDER describes a folder in a catalog. Folders allow groupings of components in a catalog (see 13.7 and Figure 8).

**Figure 8 – COMPONENT_FOLDER****7.4.11 COMPONENT_REFERENCE**

COMPONENT_REFERENCE refers to a **COMPONENT** or a **COMPONENT_FOLDER** (see 13.8 and Figure 9).

**Figure 9 – COMPONENT_REFERENCE****7.4.12 COMPONENT_RELATION**

COMPONENT_RELATION specifies a relation between **COMPONENT**s, e.g. a module and one or more other modules (see 13.9 and Figure 10).

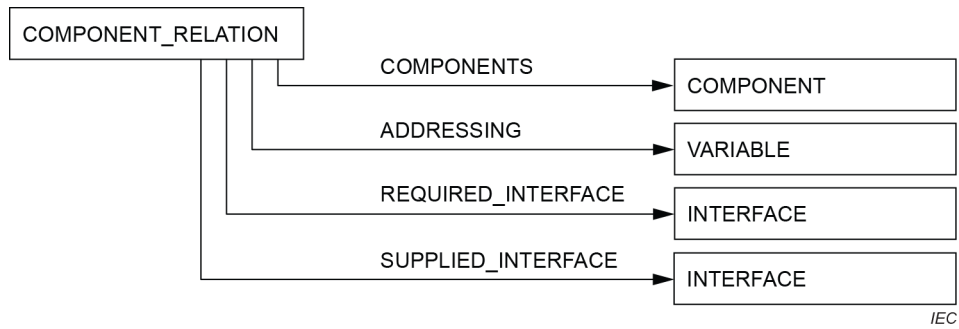


Figure 10 – COMPONENT_RELATION

7.4.13 EDIT_DISPLAY

EDIT_DISPLAY describes how the data will be represented to a user by a display device (see 11.2 and Figure 11).

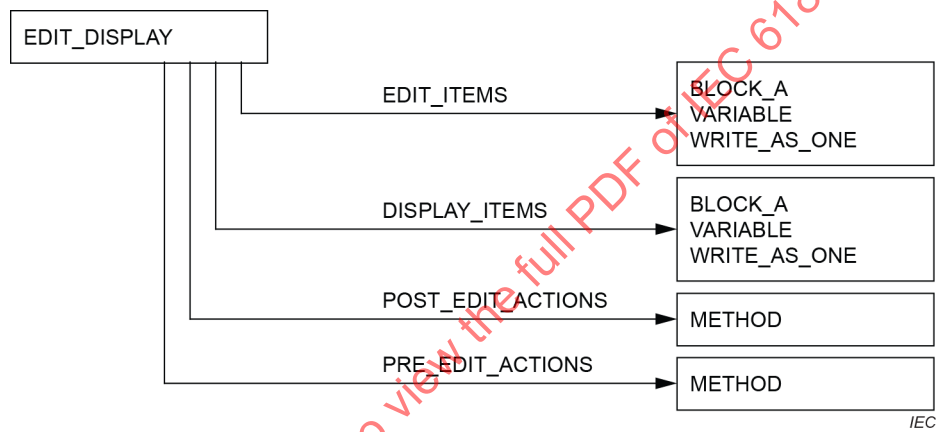


Figure 11 – EDIT_DISPLAY

7.4.14 FILE

FILE describes an area of persistent storage (see 9.6 and Figure 12).

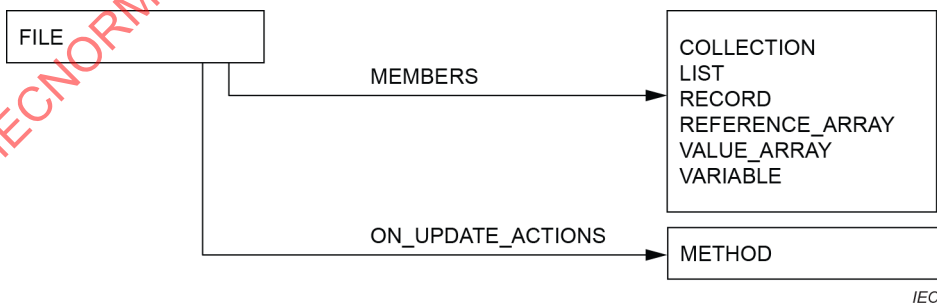
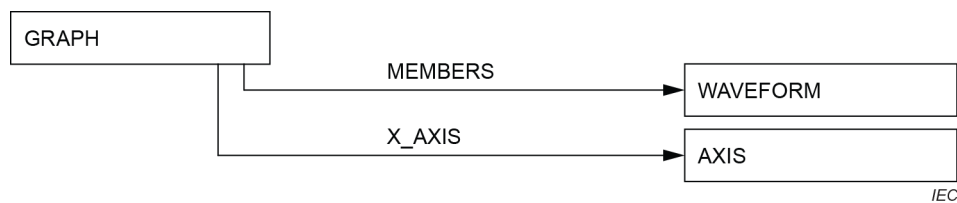


Figure 12 – FILE

7.4.15 GRAPH

GRAPH describes a graph used to display data from a device in the EDD application (see 11.8 and Figure 13).

**Figure 13 – GRAPH****7.4.16 GRID**

GRID describes a matrix used to display data from a device in the EDD application (see 11.4 and Figure 14).

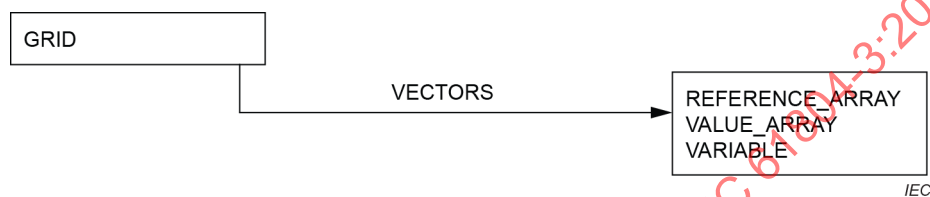
**Figure 14 – GRID****7.4.17 IMAGE**

IMAGE describes a visual image (see 11.3 and Figure 15).

**Figure 15 – IMAGE****7.4.18 IMPORT**

IMPORT is used to import an EDD and to modify its existing definitions (see 13.4).

7.4.19 INTERFACE

INTERFACE is used to specify a set of basic constructs of components that are accessible by other components (see 13.10).

7.4.20 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 13.3 and Figure 16).

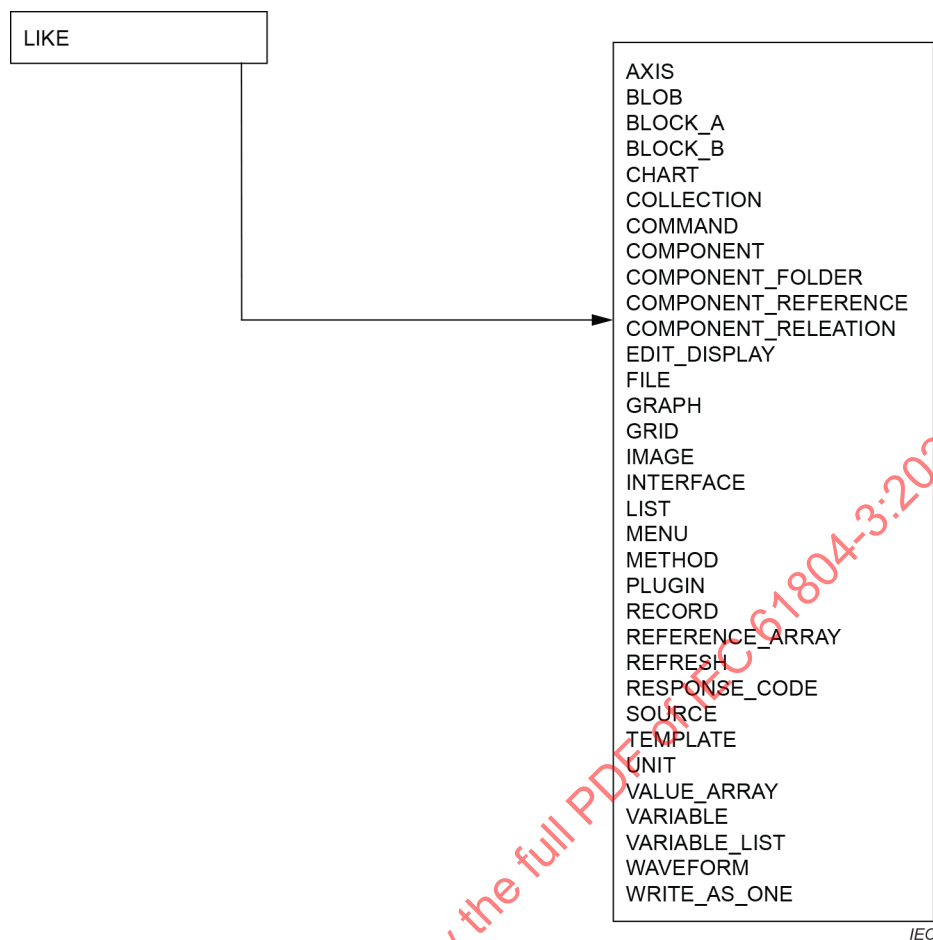


Figure 16 – LIKE

7.4.21 LIST

LIST describes a sequence of elements (see 9.4 and Figure 17).

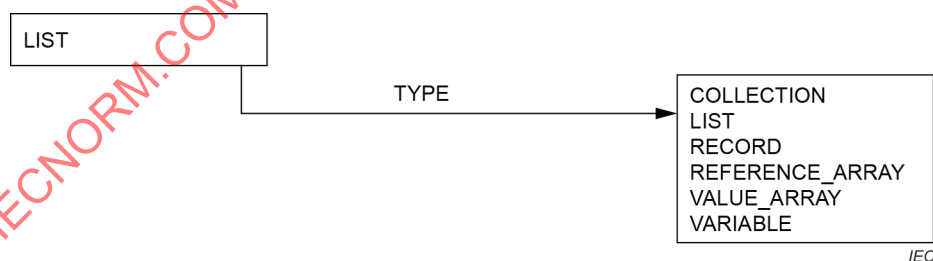


Figure 17 – LIST

7.4.22 MENU

MENU describes how the data will be presented to a user by an EDD application (see 11.1 and Figure 18).

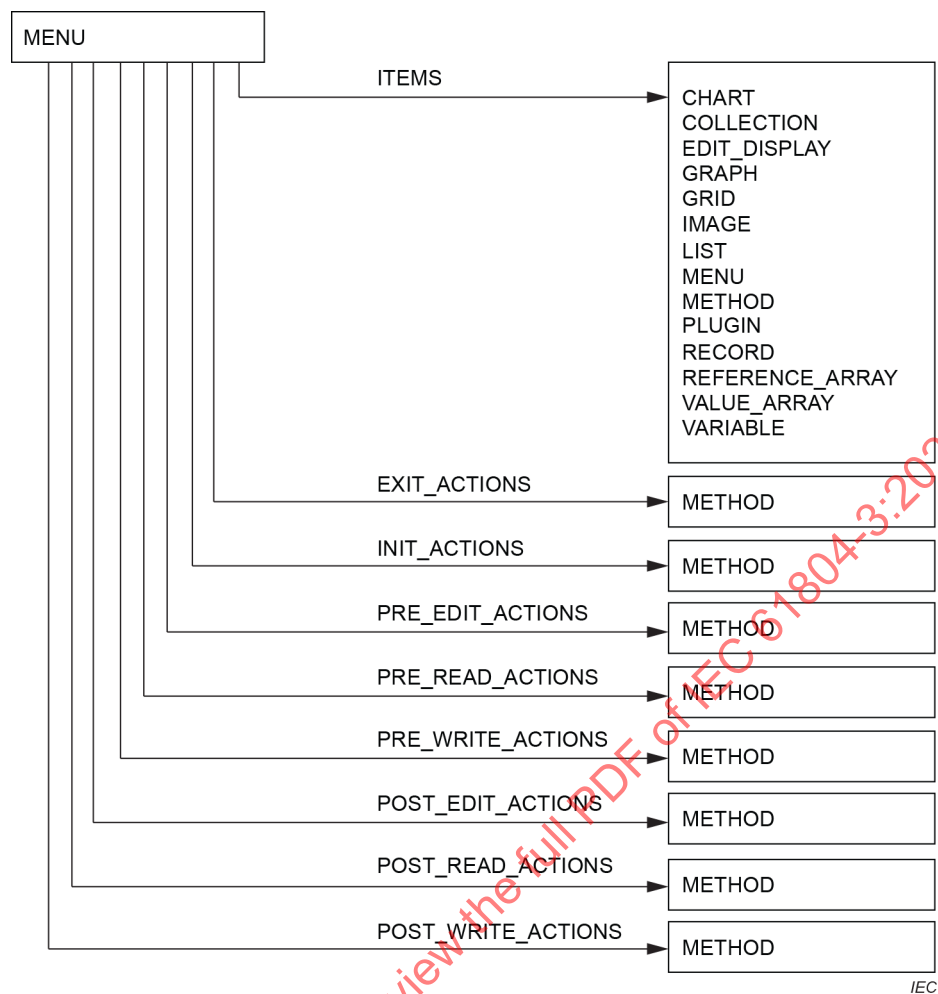


Figure 18 – MENU

7.4.23 METHOD

METHOD describes the execution of a complex sequence of event interactions that take place between system devices such as display units, configurators and devices (see Clause 12).

7.4.24 PLUGIN

PLUGIN describes a User Interface Plug-in (UIP) as specified in 11.10.

7.4.25 RECORD

RECORD describes the data contained in the device (see 10.2 and Figure 19).

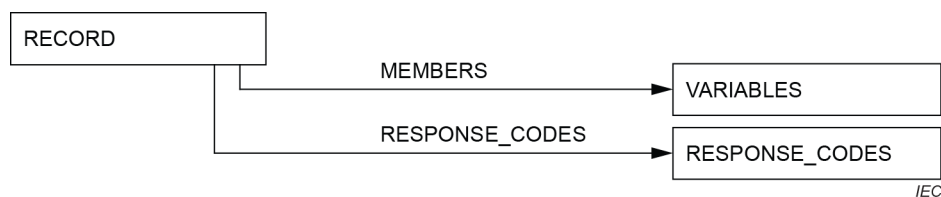


Figure 19 – RECORD

7.4.26 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 9.2 and Figure 20).

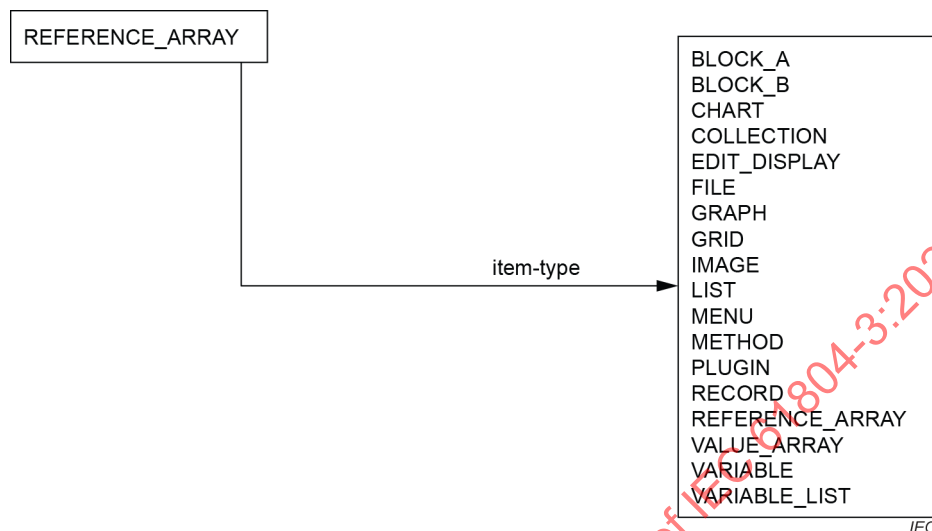


Figure 20 – REFERENCE_ARRAY

7.4.27 Relations

7.4.27.1 REFRESH

REFRESH describes relationships between LISTs, RECORDs, VALUE_ARRAYs and VARIABLES (see 9.7.1 and Figure 21).

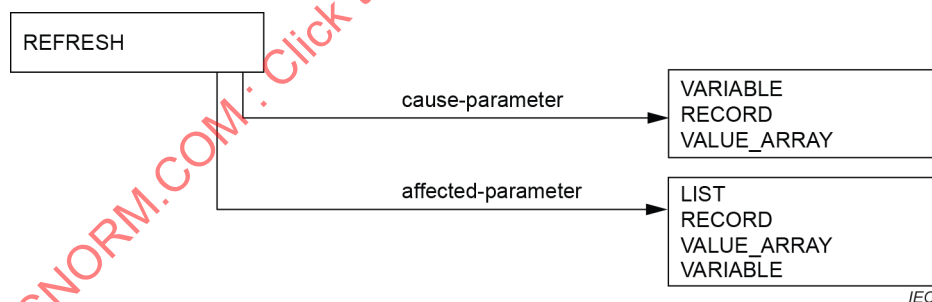


Figure 21 – REFRESH

7.4.27.2 SEMANTIC_MAP

SEMANTIC_MAP describes relationships between standardized semantic identifiers and data items (see 9.7.2 and Figure 22). Semantic identifiers are locale-independent strings typically specified in international standards like the IEC Common Data Dictionary (CDD) based on IEC 61360 (all parts) and IEC 61987 (all parts).

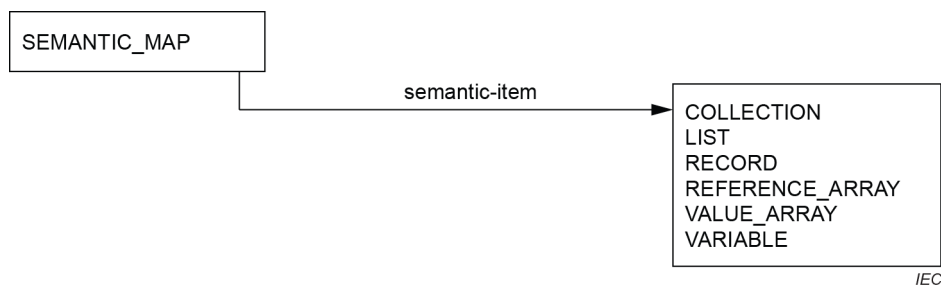


Figure 22 – SEMANTIC_MAP

7.4.27.3 UNIT

UNIT describes relationships between AXISs, VALUE_ARRAYs and VARIABLEs (see 9.7.3 and Figure 23).

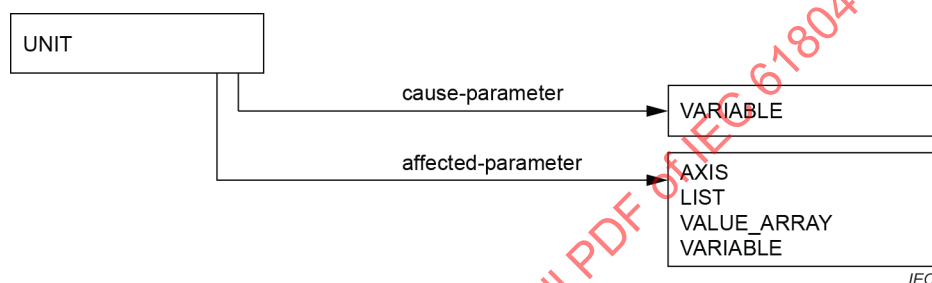


Figure 23 – UNIT

7.4.27.4 WRITE_AS_ONE

WRITE_AS_ONE describes relationships between VARIABLEs, RECORDs, and VALUE_ARRAYs (see 9.7.4 and Figure 24).



Figure 24 – WRITE_AS_ONE

7.4.28 RESPONSE_CODES

RESPONSE_CODES specifies the list of response code elements that are used by the EDD application to inform the user about the communication result, e.g. error, warning (see 10.3).

7.4.29 SOURCE

SOURCE describes a source of data values for a CHART (see 11.6 and Figure 25).

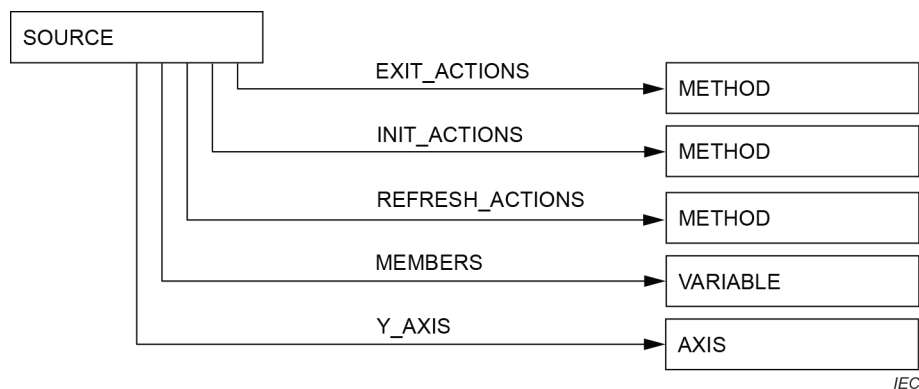


Figure 25 – SOURCE

7.4.30 TEMPLATE

TEMPLATE describes a set of parameter values that are used for offline configuration by the EDD application (see 13.5).

7.4.31 VALUE_ARRAY

VALUE_ARRAY describes logical groups of EDDL variables (see 9.3 and Figure 26).

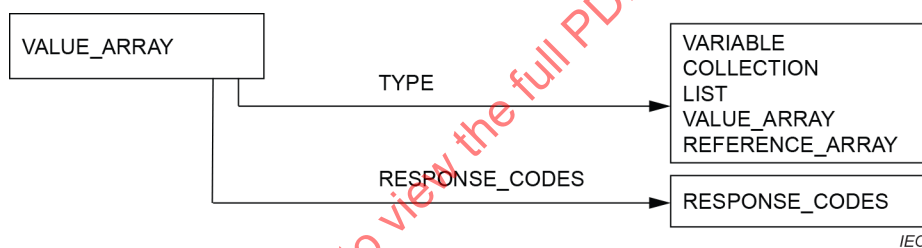


Figure 26 – VALUE_ARRAY

7.4.32 VARIABLE

VARIABLE describes the data contained in the device (see 9.1 and Figure 27).

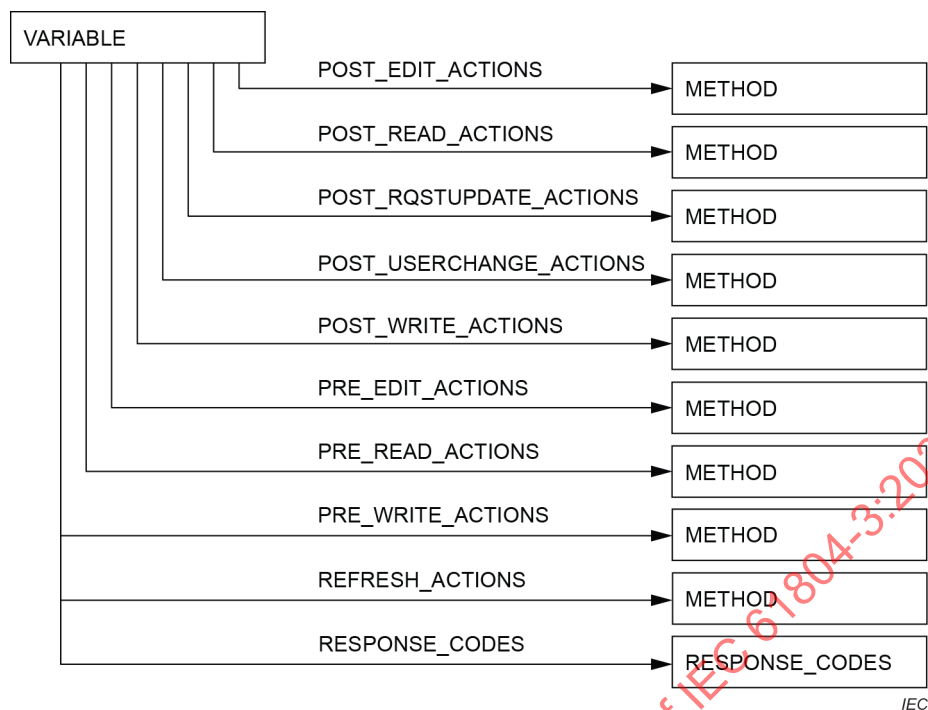


Figure 27 – VARIABLE

7.4.33 VARIABLE_LIST

VARIABLE_LIST describes logical groupings of data contained in the device that may be communicated as a list (see 10.4 and Figure 28).

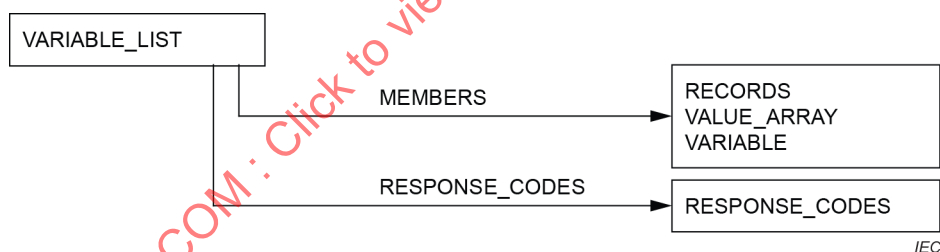


Figure 28 – VARIABLE_LIST

7.4.34 WAVEFORM

WAVEFORM describes a data set that may be displayed by a GRAPH (see 11.9 and Figure 29).

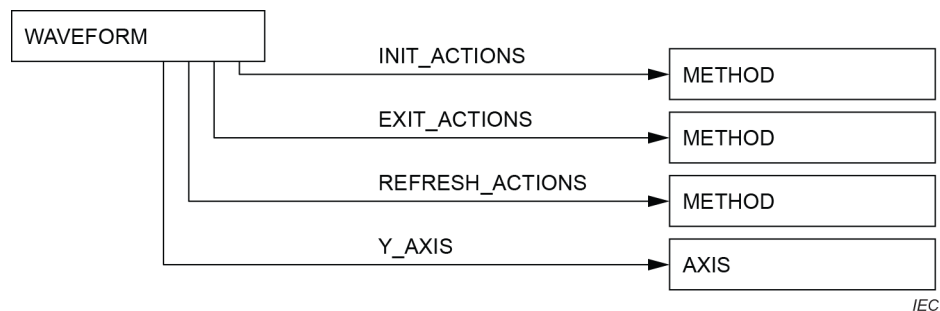


Figure 29 – WAVEFORM

7.5 Common attributes

The following common attributes are used in many basic constructs:

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

7.6 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**, which specify attributes values, which are dependent on the values of variables at run-time;
- **reference**, used throughout a device description to refer to other items within the same EDDL;
- **expression**, which is a logical or mathematical operation on variable values in order to modify attribute values of a basic construct at run-time.

7.7 Rules for instances

Each instance of a basic construct is identified by an identifier of type string. This identifier is 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 is 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.

7.8 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.

8 EDD identification information

8.1 General structure

Purpose

EDD identification information uniquely identifies the device description of a specific device type from a device manufacturer. EDD applications may read parameters from the device that correspond to these attributes and use them to identify the EDD. EDD identification information 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 *integer-constant* ,
DEVICE_REVISION *integer-constant* ,
DEVICE_TYPE [*integer-constant* | *identifier*] ,
EDD_PROFILE *integer-constant* ,
EDD_VERSION *integer-constant* ,
LAYOUT_TYPE [COLUMNWIDTH_EQUAL | COLUMNWIDTH_OPTIMIZED] ,
MANUFACTURER [*integer-constant* | *identifier*] ,
MANUFACTURER_EXT *string-constant*

The EDD identification attributes are specified in Table 6.

Table 6 – EDD identification information attributes

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
8.2	DD_REVISION	m	m	m	m	m	m	m
8.3	DEVICE_REVISION	m	m	m	m	m	m	m
8.4	DEVICE_TYPE	m	m	m	m	m	m	m
8.5	EDD_PROFILE	o	x	o	x	o	o	o
8.6	EDD_VERSION	o	x	o	x	x	o	o
8.7	LAYOUT_TYPE	o	o	o	o	o	o	o
8.8	MANUFACTURER	m	m	m	m	m	m	m
8.9	MANUFACTURER_EXT	c	x	c	x	x	c	c

8.2 DD_REVISION

Purpose

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

There can be multiple EDDs for a particular revision of a particular device, in which case the DD_REVISION distinguishes them. However, an individual EDD cannot support more than one type or revision of a device.

The manufacturer should change the revision number each time a new device description is issued for a particular DEVICE_REVISION of the device. DD revisioning rules may be further constrained by policies set forth by protocol foundations.

Lexical structure

DD_REVISION *integer-constant*

The attribute of DD_REVISION is specified in Table 7.

Table 7 – DD_REVISION attribute

Attribute	Description
integer-constant	Communication profiles: CS, GPE, HART, PB, PN Two-octet unsigned integer value that uniquely identifies the revision level of this Device Description for the device, as defined by the Manufacturer. Communication profiles: FF, ISA100 One-octet unsigned integer value that uniquely identifies the revision level of this Device Description for the device, as defined by the Manufacturer.

8.3 DEVICE_REVISION

Purpose

The DEVICE_REVISION attribute specifies a unique code for the revision of the device, as defined by the manufacturer. Device revisioning rules may be constrained by policies set forth by protocol foundations.

Lexical structure

DEVICE_REVISION *integer-constant*

The attribute of DEVICE_REVISION is specified in Table 8.

Table 8 – DEVICE_REVISION attribute

Attribute	Description
integer-constant	Communication profiles: CS, GPE, HART, PB, PN, ISA100 Two-octet unsigned integer value that uniquely identifies the revision level of the device, as defined by the Manufacturer.
integer-constant	Communication profiles: FF One-octet unsigned integer value that uniquely identifies the revision level of the device, as defined by the Manufacturer.

8.4 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 within the same manufacturer.

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

Lexical structure

DEVICE_TYPE [*integer-constant* | *identifier*]

The attributes of DEVICE_TYPE are specified in Table 9.

Table 9 – DEVICE_TYPE attribute

Attribute	Description
integer-constant	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Two-octet unsigned integer containing the code for a specific device type or group of device types.
identifier	Communication profiles: CS, GPE, HART, PB, PN 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.	

8.5 EDD_PROFILE

Purpose

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

Lexical structure

EDD_PROFILE *integer-constant*

The attribute of EDD_PROFILE is specified in Table 10.

Table 10 – EDD_PROFILE attribute

Attribute	Description
integer-constant	One-octet unsigned integer containing the code for the specific profiles. The following codes are currently defined: 0x01 Profile for HART 0x02 Profile for FOUNDATION Fieldbus 0x03 Profile for PROFIBUS 0x04 Profile for PROFINET 0x05 Profile for ISA100_Wireless 0x06 Profile for Generic Protocol Extension 0x07 Profile for Communication Server

NOTE Although not all communication profiles support the EDD_PROFILE attribute, the values assigned to all profiles are listed in Table 10 for completeness.

8.6 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 document.

Lexical structure

EDD_VERSION *integer-constant*

The attribute of EDD_VERSION is specified in Table 11.

Table 11 – EDD_VERSION attribute

Attribute	Description
integer	Two-octet unsigned integer containing the number 1.

8.7 LAYOUT_TYPE

Purpose

The LAYOUT_TYPE attribute specifies how the EDD application shall render the entire graphical user interface defined in this EDD.

Lexical structure

LAYOUT_TYPE [COLUMNWIDTH_EQUAL | COLUMNWIDTH_OPTIMIZED]

The attributes of LAYOUT_TYPE are specified in Table 12.

Table 12 – LAYOUT_TYPE attribute

Attribute value	Description
COLUMNWIDTH_EQUAL	All MENUs whose STYLE attribute is WINDOW, DIALOG, PAGE, or GROUP shall be rendered with equal width columns. Communication profiles: FF, HART, ISA100 The default value of LAYOUT_TYPE is COLUMNWIDTH_EQUAL
COLUMNWIDTH_OPTIMIZED	All MENUs whose STYLE attribute is WINDOW, DIALOG, PAGE, or GROUP shall be rendered with optimized width columns. Optimized width columns shall be sized according to the widest element in the column across all rows with the same number of columns. Communication profiles: CS, GPE, PB, PN The default value of LAYOUT_TYPE is COLUMNWIDTH_OPTIMIZED

8.8 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 [*integer-constant* | *identifier*]

The attributes of MANUFACTURER are specified in Table 13.

Table 13 – MANUFACTURER attribute

Attribute	Description
integer-constant	Communication profiles: CS, GPE, HART, PB, PN Unsigned integer containing the code for a specific manufacturer. Communication profiles: CS, GPE, PB, PN If the value of the integer is 0xFFFFF, the lexical structure "MANUFACTURER_EXT" shall be present and identifies the manufacturer.
integer-constant	Communication profiles: FF 24-bit unsigned integer containing the code for a specific manufacturer.
integer-constant	Communication profiles: ISA100 32-bit unsigned integer containing the code for a specific manufacturer. The most significant byte of the MANUFACTURER value shall be fixed to 0.
identifier	Communication profiles: CS, GPE, HART, PB, PN 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.	

8.9 MANUFACTURER_EXT

Purpose

The MANUFACTURER_EXT attribute is 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-constant*

The attribute of MANUFACTURER_EXT is specified in Table 14.

Table 14 – MANUFACTURER_EXT attribute

Attribute	Description
string-constant	String containing the information for a unique worldwide manufacturer identification.

9 Data modelling

9.1 VARIABLE

9.1.1 General structure

Purpose

VARIABLE describes parameters contained in a device or in the EDD application. These data items include process data, status, and configuration related properties. The VARIABLE

construct provides a wide range of descriptive properties allowing any data item to be fully described.

Lexical structure

```

VARIABLE identifier
{
    LABEL string-value;
    HELP string-value;
    CLASS variable-class-specifier [&variable-class-specifier]*;
    CONSTANT_UNIT string-value;
    DEFAULT_VALUE default-value-specifier;
    HANDLING handling-specifier;
    HEIGHT height-width-specifier;
    INITIAL_VALUE initial-value-specifier
    POST_EDIT_ACTIONS { actions-specifier }
    POST_READ_ACTIONS { actions-specifier }
    POST_RQSTUPDATE_ACTIONS { actions-specifier }
    POST_USERCHANGE_ACTIONS { actions-specifier }
    POST_WRITE_ACTIONS { actions-specifier }
    PRE_EDIT_ACTIONS { actions-specifier }
    PRE_READ_ACTIONS { actions-specifier }
    PRE_WRITE_ACTIONS { actions-specifier }
    PRIVATE boolean-specifier;
    REFRESH_ACTIONS { actions-specifier }
    RESPONSE_CODES response-codes-specifier
    TYPE variable-type;
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
    WIDTH height-width-specifier;
    WRITE_MODE write-mode-specifier;
}

```

The attributes of VARIABLE are specified in Table 15.

Table 15 – VARIABLE attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	VARIABLE		✓	✓	✓	✓	✓	✓	✓
9.1.2	CLASS		m	m	m	o	o	m	m
9.1.3	CONSTANT_UNIT		o	o	o	o	o	o	o
9.1.4	DEFAULT_VALUE		o	o	o	o	o	o	o
14.6	HANDLING		o	o	o	o	o	o	o
14.7	HEIGHT		o	o	o	o	o	o	o
14.8	HELP		o	o	o	o	o	o	o
9.1.5	INITIAL_VALUE		o	x	o	x	x	o	o
14.9	LABEL		o	m	o	o	m	o	o
9.1.6	POST_EDIT_ACTIONS		o	o	o	o	o	o	o
9.1.7	POST_READ_ACTIONS		o	o	o	o	o	o	o
9.1.8	POST_RQSTUPDATE_ACTIONS		x	x	x	o	x	x	x
9.1.9	POST_USERCHANGE_ACTIONS		x	x	x	o	x	x	x
9.1.10	POST_WRITE_ACTIONS		o	o	o	o	o	o	o
9.1.11	PRE_EDIT_ACTIONS		o	o	o	o	o	o	o
9.1.12	PRE_READ_ACTIONS		o	o	o	o	o	o	o

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
9.1.13		PRE_WRITE_ACTIONS	o	o	o	o	o	o	o
14.13		PRIVATE	o	o	o	o	o	o	o
9.1.14		REFRESH_ACTIONS	o	o	o	o	o	o	o
14.15		RESPONSE_CODES	x	o	x	x	o	x	x
9.1.15		TYPE	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o
14.19		WIDTH	o	o	o	o	o	o	o
14.20		WRITE_MODE	x	o	x	x	o	x	x

9.1.2 CLASS

Purpose

The CLASS attribute specifies how the VARIABLE is used by the device and the EDD application for organization purposes and display. Classes are used to model the application of the EDD item (e.g. it indicates the item is used for diagnosing the status of the device). In addition, an EDD item may be provided to assist the EDD application to access the device. In this case the EDD item is class LOCAL and does not actually exist in the device. CLASS shall not contain a conditional.

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 variable-class-specifier [&variable-class-specifier]*;
variable-class-specifier := ALARM | ANALOG_INPUT | ANALOG_OUTPUT |
COMPUTATION | CONSTANT | CONTAINED | CORRECTION | DEVICE | DIAGNOSTIC |
DIGITAL_INPUT | DIGITAL_OUTPUT | DISCRETE | DISCRETE_INPUT |
DISCRETE_OUTPUT | DYNAMIC | FREQUENCY | FREQUENCY_INPUT | FREQUENCY_OUTPUT
| HART | INPUT | IS_CONFIG | LOCAL | LOCAL_A | LOCAL_B | LOCAL_DISPLAY |
MODE | OPERATE | OPTIONAL | OUTPUT | SERVICE | SPECIALIST | TEMPORARY |
TUNE

```

The attributes of CLASS are specified in Table 16.

Table 16 – CLASS attributes

Attribute	Description	Communication profiles
ALARM	The VARIABLE contains alarm limits.	CS, FF, GPE, HART, ISA100, PB, PN
ANALOG_INPUT	The VARIABLE is part of an analog input block.	CS, GPE, PB, PN
ANALOG_OUTPUT	The VARIABLE is part of an analog output block.	CS, GPE, HART, PB, PN
COMPUTATION	The VARIABLE is part of a computation block.	CS, GPE, HART, PB, PN

Attribute	Description	Communication profiles
CONSTANT	The VARIABLE value is unchanged throughout the life time of the device. The value of these VARIABLES shall be preserved when a device undergoes a warm restart, power fail and even when device resets to factory defaults. CLASS CONSTANT shall not be combined with DYNAMIC.	CS, FF, GPE, HART, ISA100, PB, PN
CONTAINED	The VARIABLE represents the physical characteristics of the device. The VARIABLE represents a block parameter that cannot be referenced by another block input or set by a block output.	CS, FF, GPE, PB, PN
CORRECTION	The VARIABLE is part of the correction block.	CS, GPE, HART, PB, PN
DEVICE	The VARIABLE represents the physical characteristics of the device.	CS, GPE, HART, PB, PN
DIAGNOSTIC	The VARIABLE indicates the device status.	CS, FF, GPE, HART, ISA100, PB, PN
DIGITAL_INPUT	The VARIABLE is part of a digital input block.	CS, GPE, PB, PN
DIGITAL_OUTPUT	The VARIABLE is part of a digital output block.	CS, GPE, PB, PN
DISCRETE	The VARIABLE is part of a discrete block.	HART
DISCRETE_INPUT	The VARIABLE is part of a discrete input block.	CS, GPE, PB, PN
DISCRETE_OUTPUT	The VARIABLE is part of a discrete output block.	CS, GPE, PB, PN
DYNAMIC	The VARIABLE is modified by the device without stimulus from the network. Write-able variables of CLASS DYNAMIC may be edited. When edited, the value displayed stops updating and the result of the edit is maintained in the display value until committed to the device. After committing the variable value once again starts updating continuously. CLASS DYNAMIC shall not be combined with CONSTANT.	CS, FF, GPE, HART, ISA100, PB, PN
FREQUENCY	The VARIABLE is part of a frequency block.	HART
FREQUENCY_INPUT	The VARIABLE is part of a frequency input block.	CS, GPE, HART, PB, PN
FREQUENCY_OUTPUT	The VARIABLE is part of a frequency output block.	CS, GPE, HART, PB, PN
HART	The VARIABLE is part of the HART block which characterizes the HART interface. There is only one HART block.	CS, HART
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.	CS, FF, GPE, HART, ISA100, PB, PN
IS_CONFIG	A modification of the VARIABLE results in setting the revision counter or configuration-changed bit, as applicable.	CS, FF, GPE, HART, PB, PN
LOCAL	Communication profiles: HART, FF, ISA100 VARIABLEs of CLASS LOCAL behave as in LOCAL_A. Communication profiles: CS, GPE, PB, PN VARIABLEs of CLASS LOCAL behave as in LOCAL_B.	CS, FF, GPE, HART, ISA100, PB, PN

Attribute	Description	Communication profiles
LOCAL_A	The VARIABLE is locally used by the EDD application. Local VARIABLES are not stored in a device, but they can be communicated to or from a device. CLASS LOCAL_A variables shall not belong to any other class, except TEMPORARY. VARIABLEs of CLASS LOCAL_A are automatically committed to the active data set after modification. Active data set is either the online or offline data set. All caches based on the active data set are also synchronized. Communication profiles: HART When a VARIABLE of CLASS LOCAL_A is used in a COMMAND TRANSACTION, the variable shall include the INFO qualifier.	CS, FF, GPE, HART, ISA100, PB, PN
LOCAL_B	The VARIABLE is locally used by the EDD application. Local VARIABLES are not stored in a device, but they can be communicated to or from a device. CLASS LOCAL_B variables shall not belong to any other class, except TEMPORARY. VARIABLEs of CLASS LOCAL_B are not committed automatically to the active data set after modification. They are committed at the same time as device variables Communication profiles: HART When a VARIABLE of CLASS LOCAL_B is used in a COMMAND TRANSACTION, the variable shall include the INFO qualifier.	CS, FF, GPE, HART, ISA100, PB, PN
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.	CS, GPE, HART, PB, PN
MODE	The VARIABLE controls an operating mode of the device.	HART
OPERATE	The VARIABLE is used to control a block's operation (for instance, set point).	CS, FF, GPE, PB, PN
OPTIONAL	The VARIABLE may not be present in the device. A device profile defines variables as mandatory or optional. If a device supports a feature it shall support it in the way defined by the optional variable. If a device does not support an optional variable the EDD application shall not display the value of this variable. The EDD application may hide the whole variable in that case. If a method reads a not supported optional variable the VARIABLE_STATUS is set to VARIABLE_STATUS_NOT_SUPPORTED and the value of the variable does not exist. Before the method accesses the variable value, the method shall check the VARIABLE_STATUS otherwise the method aborts. If a method writes a not supported optional variable the VARIABLE_STATUS is set to VARIABLE_STATUS_NOT_SUPPORTED.	CS, GPE, PB, PN
OUTPUT	The VARIABLE is part of the output block. The values of output VARIABLEs may be accessed by another block input.	CS, FF, GPE, ISA100, PB, PN
SERVICE	The VARIABLE is used when performing routine maintenance.	CS, FF, GPE, HART, ISA100, PB, PN

Attribute	Description	Communication profiles
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 independent 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. VARIABLEs of CLASS SPECIALIST can still be accessed and changed from METHODS.	CS, GPE, HART, ISA100, PB, PN
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.	CS, GPE, PB, PN
TUNE	The VARIABLE is used to tune the algorithm of a block.	CS, FF, GPE, HART, ISA100, PB, PN
Communication profiles: FF One and only one of the classes: INPUT, OUTPUT, CONTAINED.		
Communication profiles: ISA100 CONSTANT and DYNAMIC can be combined with zero or more of: INPUT, OUTPUT, ALARM, DIAGNOSTIC, SERVICE, SPECIALIST, TUNE When neither DYNAMIC nor CONSTANT is defined, then the VARIABLE only needs to be refreshed when a change is detected.		
Communication profiles: HART If not CLASS LOCAL or CONSTANT then can be combined with DYNAMIC, DIAGNOSTIC, SERVICE, and IS_CONFIG. The default is no CLASS. In this case, the VARIABLE is not DYNAMIC, CONSTANT, nor LOCAL, so the VARIABLE is read from the device but only needs to be refreshed when a change is detected.		

9.1.3 CONSTANT_UNIT

Purpose

The CONSTANT_UNIT attribute is used if a VARIABLE has a units code which never changes (e.g. damping is always in seconds). The CONSTANT_UNIT is specified as a string that will be displayed along with the VARIABLE's value. A VARIABLE without a constant unit either has no units associated with it or the units are not constant. When the engineering units can vary they shall be represented using a UNIT relation.

Lexical structure

CONSTANT_UNIT *string-value* | *string-value<cond>*;

The attribute of CONSTANT_UNIT is specified in Table 17.

Table 17 – CONSTANT_UNIT attribute

Value type	Description
string-value	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Units code string to be displayed.
string-value<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Condition that specifies the units code string to be displayed

9.1.4 DEFAULT_VALUE

Purpose

The DEFAULT_VALUE attribute specifies the default setting for the VARIABLE. DEFAULT_VALUE allows the EDD to specify a logically consistent and reasonable value the EDD application may use when the device is not physically in communication with the EDD application (e.g. when generating a new device configuration offline). If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as the INITIAL_VALUE.

The type of the expression shall be consistent with the type of the VARIABLE. The expression shall evaluate to a constant without requiring communication with the actual device. The DEFAULT_VALUE may also be specified via the TYPE attribute of the VARIABLE.

For arithmetic types, the DEFAULT_VALUE is unscaled.

Communication profiles: HART

When a DEFAULT_VALUE is not specified, the VARIABLE shall assume the value listed in the column "Initial value" in Table B.2 until edited by the user or until communication with the device changes the VARIABLE's value. If the initial value from Table B.2 is not within the MIN_VALUE and MAX_VALUE range it will be set to the nearest value that is in range.

Lexical structure

Communication profiles: FF, ISA100, HART

DEFAULT_VALUE *expression*;

Communication profiles: CS, GPE, PB, PN

DEFAULT_VALUE *expression*<cond>;

The type of the expression shall be consistent with the type of the VARIABLE.

9.1.5 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.

The column "Initial value" in Table B.2 shows the initial values depending on the type of the VARIABLE.

Lexical structure

INITIAL_VALUE *expression*;

The type of the expression shall be consistent with the type of the VARIABLE.

9.1.6 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 the editing is cancelled by the user, the POST_EDIT_ACTIONS shall not be executed. If a METHOD exits for an unplanned reason, the following METHODS are not executed. If one of the POST_EDIT_ACTIONS METHODS aborts, then the data cache has already been updated with the new value and is unaffected by the abort.

POST_EDIT_ACTIONS should be used to display information or warnings or to manipulate variables. The methods are called after the value of a variable is changed.

If a METHOD of POST_EDIT_ACTIONS aborts, the EDD application should give the user an indication of the abnormal process.

Lexical structure

```
POST_EDIT_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                    [DEFINITION { compound-statement }]<cond>
```

The attributes of POST_EDIT_ACTIONS are specified in Table 18.

Table 18 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS attributes

Attribute	Description
reference	Reference to a METHOD instance or a METHOD instance with arguments.
DEFINITION	Communication profiles: CS, GPE, PB, PN Lexical element (see 14.4).

9.1.7 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. POST_READ_ACTIONS are performed immediately after the responses are received and before any additional communication requests are dispatched. 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. If one of the POST_READ_ACTIONS METHODS aborts, the EDD application shall mark the variable as not being read.

Lexical structure

```
POST_READ_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                    [DEFINITION { compound-statement }]<cond>
```

The attributes of POST_READ_ACTIONS are specified in Table 18.

9.1.8 POST_RQSTUPDATE_ACTIONS

Purpose

POST_RQSTUPDATE_ACTIONS shall only be supported by device simulators. POST_RQSTUPDATE_ACTIONS, when present, shall be ignored in all EDD applications except device simulators.

The POST_RQSTUPDATE_ACTIONS attribute specifies METHODS that shall be executed when the VARIABLE's value is written via communication from the EDD application, i.e. when the VARIABLE is updated via a request packet (whether changed or not). This means that each variable's post-request-update actions will occur before the variables following it in the receive packet have been updated. The actions shall be executed and completed prior to the simulator generating a response packet. 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<cond> [, reference<cond>]*
}
```

The attribute of POST_RQSTCHANGE_ACTIONS is specified in Table 19.

Table 19 – POST_USERCHANGE_ACTIONS, POST_RQSTUPDATE_ACTIONS attributes

Attribute	Description
reference	Reference to a METHOD.

9.1.9 POST_USERCHANGE_ACTIONS

Purpose

POST_USERCHANGE_ACTIONS shall only be supported by device simulators. POST_USERCHANGE_ACTIONS, when present, shall be ignored in all EDD applications except 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. This attribute operates like POST_EDIT_ACTIONS, except they will only be executed if the user changes the value in 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_USERCHANGE_ACTIONS

```
{
  reference<cond> [, reference<cond>]*
}
```

The attribute of POST_USERCHANGE_ACTIONS is specified in Table 19.

9.1.10 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. POST_WRITE_ACTIONS are performed immediately after the responses are received and before any additional communication requests are dispatched. 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. If one of the POST_WRITE_ACTIONS METHODS aborts, the EDD application shall mark the variable as not being written.

Lexical structure

```
POST_WRITE_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*] |
                    [DEFINITION { compound-statement }]<cond>
```

The attributes of POST_WRITE_ACTIONS are specified in Table 18.

9.1.11 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.

PRE_EDIT_ACTIONS should be used to display information or warnings to the user. They should not be used for any manipulations of variables. The methods are called only if the user wants to edit the variable.

If a METHOD of PRE_EDIT_ACTIONS aborts the EDD application should give the user an indication of the abnormal process and the edit shall be cancelled.

Lexical structure

```
PRE_EDIT_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*] |
                    [DEFINITION { compound-statement }]<cond>
```

The attributes of PRE_EDIT_ACTIONS are specified in Table 18.

9.1.12 PRE_READ_ACTIONS

Purpose

The PRE_READ_ACTIONS attribute specifies METHODS that shall be executed before the VARIABLE is read. PRE_READ_ACTIONS are performed immediately before the communication requests are dispatched. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the subsequent METHODS are not executed and the read operation shall not be performed and fails.

Lexical structure

```

PRE_READ_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                     [DEFINITION { compound-statement }]<cond>

```

The attributes of PRE_READ_ACTIONS are specified in Table 18.

9.1.13 PRE_WRITE_ACTIONS**Purpose**

The PRE_WRITE_ACTIONS attribute specifies METHODS that shall be executed before the VARIABLE is written to the device. PRE_WRITE_ACTIONS are performed immediately before the communication requests are dispatched. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the subsequent METHODS are not executed and the write operation shall not be performed and fails.

Lexical structure

```

PRE_WRITE_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                     [DEFINITION { compound-statement }]<cond>

```

The attributes of PRE_WRITE_ACTIONS are specified in Table 18.

9.1.14 REFRESH_ACTIONS**Purpose**

Communication profiles: HART

REFRESH_ACTIONS are executed just prior to displaying or updating the VARIABLE on the user interface. REFRESH_ACTIONS are executed within methods (e.g. when a VARIABLE is displayed in PUT_MESSAGE, etc.). If a REFRESH_ACTION aborts, the value shall not be displayed.

Communication profiles: CS, GPE, PB, PN, FF, ISA100

The REFRESH_ACTIONS attribute specifies METHODS that shall be executed to calculate the value of the VARIABLE based on other VARIABLES. 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 methods shall be called every time before the VARIABLE needs to be evaluated for conditional expressions, for display purpose, etc.

REFRESH_ACTIONS shall not be used to modify other variables or to display anything.

Lexical structure

```

REFRESH_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                     [DEFINITION { compound-statement }]<cond>

```

The attributes of REFRESH_ACTIONS are specified in Table 18.

9.1.15 TYPE

9.1.15.1 General structure

Purpose

The TYPE attribute describes the format of the VARIABLES value.

Lexical structure

TYPE *variable-type*
variable-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 20.

Table 20 – TYPE overview

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.15.10	ASCII	✓	✓	✓	✓	✓	✓	✓
9.1.15.12	BITSTRING	✓	✓	✓	✗	✓	✓	✓
9.1.15.6	BIT_ENUMERATED	✓	✓	✓	✓	✓	✓	✓
9.1.15.13	BOOLEAN	✓	✓	✓	✗	✓	✓	✓
9.1.15.5	DATE	✓	✗	✓	✓	✗	✓	✓
9.1.15.5	DATE_AND_TIME	✓	✓	✓	✗	✗	✓	✓
9.1.15.4	DOUBLE	✓	✓	✓	✓	✓	✓	✓
9.1.15.5	DURATION	✓	✓	✓	✗	✓	✓	✓
9.1.15.7	ENUMERATED	✓	✓	✓	✓	✓	✓	✓
9.1.15.10	EUC	✓	✓	✓	✗	✓	✓	✓
9.1.15.4	FLOAT	✓	✓	✓	✓	✓	✓	✓
9.1.15.8	INDEX	✓	✓	✓	✓	✓	✓	✓
9.1.15.4	INTEGER	✓	✓	✓	✓	✓	✓	✓
9.1.15.9	OBJECT_REFERENCE	✓	✗	✓	✗	✗	✓	✓
9.1.15.11	OCTET	✓	✓	✓	✗	✓	✓	✓
9.1.15.10	PACKED_ASCII	✓	✗	✓	✓	✗	✓	✓
9.1.15.10	PASSWORD	✓	✓	✓	✓	✓	✓	✓
9.1.15.5	TIME	✓	✓	✓	✗	✓	✓	✓
9.1.15.5	TIME_VALUE(4)	✓	✗	✓	✓	✓	✓	✓
9.1.15.5	TIME_VALUE(8)	✓	✓	✓	✗	✗	✓	✓
9.1.15.4	UNSIGNED_INTEGER	✓	✓	✓	✓	✓	✓	✓
9.1.15.10	VISIBLE	✓	✓	✓	✗	✓	✓	✓

9.1.15.2 MIN_VALUE, MAX_VALUE

Purpose

MIN_VALUE and MAX_VALUE specify the lower bound and upper bound of values to which a VARIABLE shall be set. If the variable is dynamic, see CLASS attribute, a device can set the value of the variable outside the range specified by its MIN_VALUE and MAX_VALUE

attributes. The minimum and maximum values are compared to the unscaled value. Support of MIN_VALUE and MAX_VALUE attributes depends on the TYPE attribute. Subsections of each of the variable types below will identify the lexical structure and support for each TYPE.

A VARIABLE may have more than one minimum or maximum value, for instance, to specify more than one range of validity (e.g. the variable may have a range just above zero and just below zero but not exactly at zero). When there are multiple minimum or maximum values, additional integers are used to number each MIN_VALUE or 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/MAX_VALUE can also be built by an expression (see Clause 18).

Communication profiles: FF, HART, ISA100

Values returned from the device shall be accepted and displayed even if they are outside the MIN_VALUE and MAX_VALUE

Lexical structure

`[MAX_VALUE | MIN_VALUE][n]? expression<cond>;`

The type of the expression shall be consistent with the type of the VARIABLE.

The attribute of MIN_VALUE and MAX_VALUE is specified in Table 21.

Table 21 – MIN_VALUE, MAX_VALUE attribute

Value type	Description
n	An increasing integer constant from 1 to 9, which identifies the MIN_VALUE or MAX_VALUE if more than one is present.

Usage

The minimum and maximum values for a variable may be referenced from elsewhere in the EDD as specified in 19.2.

For following example specifies two ranges: one from –10 to –5 and another from 5 to 10.

```
MIN_VALUE1 -10;
MAX_VALUE1 -5;
MIN_VALUE2 5;
MAX_VALUE2 10;
```

9.1.15.3 value / description / help

Purpose

The attribute value contains a constant that specifies the value. The type depends on the data type specified with the TYPE attribute. If the attribute value is defined a description attribute is required. Equal values are not allowed. Support of value, description, and help attributes depends on the TYPE attribute. Subsections of each of the variable types below will identify the lexical structure and support for each TYPE.

The attribute description contains a short description of the item.

The attribute help specifies help text for the item.

Lexical structure

```
value:= [double-constant | float-constant | integer-constant | unsigned-integer-constant | string-constant]
description:= string-constant
help:= string-constant
```

9.1.15.4 Arithmetic types

9.1.15.4.1 General structure

Purpose

VARIABLEs of TYPE FLOAT and DOUBLE are single-precision basic format and double-precision basic format floating-point numbers, respectively, as defined in IEEE 754. 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 double-value;
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
  INITIAL_VALUE double-constant;
  [MAX_VALUE[n]? double-value;]*
  [MIN_VALUE[n]? double-value;]*
  SCALING_FACTOR double-value;
  [{value , description , help }]<cond>]*
}]?
```

Lexical structure – FLOAT

FLOAT

```
[{
  DEFAULT_VALUE float-value;
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
  INITIAL_VALUE float-constant;
  [MAX_VALUE[n]? float-value;]*
  [MIN_VALUE[n]? float-value;]*
  SCALING_FACTOR float-value;
  [{value , description , help }]<cond>]*
}]?
```

Lexical structure – INTEGER

INTEGER (size)

```
[{
  DEFAULT_VALUE integer-value;
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
  INITIAL_VALUE integer-constant;
  [MAX_VALUE[n]? integer-value;]*
  [MIN_VALUE[n]? integer-value;]*
  SCALING_FACTOR integer-value;
  [{value , description , help }]<cond>]*
}]?
```

Lexical structure – UNSIGNED_INTEGER

```

UNSIGNED_INTEGER (size)
[ {
    DEFAULT_VALUE unsigned-integer-value;
    DISPLAY_FORMAT string-value;
    EDIT_FORMAT string-value;
    INITIAL_VALUE unsigned-integer-constant;
    [MAX_VALUE [n]? unsigned-integer-value;]*
    [MIN_VALUE [n]? unsigned-integer-value;]*
    SCALING_FACTOR unsigned-integer-value;
    [{value , description , help }]<cond>]*
}]?

```

Table 22 – TYPE attributes for DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.3	description	o	x	o	x	x	o	o
9.1.15.4.2	DISPLAY_FORMAT	o	o	o	o	o	o	o
9.1.15.4.3	EDIT_FORMAT	o	o	o	o	o	o	o
9.1.15.3	help	o	x	o	x	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.2	MAX_VALUE	o	o	o	o	o	o	o
9.1.15.2	MIN_VALUE	o	o	o	o	o	o	o
9.1.15.4.4	SCALING_FACTOR	o	o	o	o	o	o	o
9.1.15.4.5	size	o	o	o	o	o	o	o
9.1.15.3	value	o	x	o	x	x	o	o

9.1.15.4.2 DISPLAY_FORMAT

Purpose

Specifies how the value of the VARIABLE is displayed. The DISPLAY_FORMAT string shall be defined according to the function printf of the C programming language without the leading '%' qualifier (see ISO/IEC 9899:1999). When a DISPLAY_FORMAT attribute is not specified, the arithmetic type of the variable determines the default format string as described in Table 23.

Table 23 – Default display format conversion type

Arithmetic type	Display format
INTEGER	d
UNSIGNED_INTEGER	u
FLOAT	g
DOUBLE	g

Communication profiles: CS, GPE, PB, PN

When the default format specifier from Table 23 is used, the default width and precision are limited only by the arithmetic type of the VARIABLE.

Communication profiles: HART, FF, ISA100

When the default format specifier from Table 23 is used, the default width and precision as described in Table 24 are also used.

Table 24 – Default width/precision based on VARIABLE and format conversion type

Type	Conversion type	Size ^a	Width	Precision
FLOAT	f, F, e, E, g, G	n/a	12	5
	d, l, o, u	n/a	11	0
	x	n/a	8	0
DOUBLE	f, F, e, E, g, G	n/a	18	8
	d, l, o, u	n/a	15	0
	x	n/a	16	0
INTEGER UNSIGNED_INTEGER	f, F, e, E, g, G	n/a	12	6
	d, l, u	1	4	0
		2	6	0
		4	11	0
		8	20	0
	o	1	3	0
		2	6	0
		4	11	0
		8	22	0
	x	1	2	0
		2	4	0
		4	8	0
		8	16	0

^a n/a = not applicable

Lexical structure

DISPLAY_FORMAT *string-value*<cond>;

9.1.15.4.3 EDIT_FORMAT

Purpose

Specifies how the value of the VARIABLE is edited. The EDIT_FORMAT string shall be defined according to the function scanf of the C programming language (see ISO/IEC 9899:1999). An edit format does not contain the leading '%' qualifier and only controls what is scanned when parsing a user edited value. When an EDIT_FORMAT attribute is not specified, the DISPLAY_FORMAT used for the variable shall be used.

Lexical structure

EDIT_FORMAT *string-value*<cond>;

The edit format consists of an optional width numeric specifier, an optional decimal point and precision specifier, followed by a type specifier.

flags width* [.precision]* type*

The flags may contain several specifiers including the hash (#) and zero (0) specifiers. Their use is the same as specified in the ANSI C function printf.

The width specifier is a decimal integer that defines the maximum number of characters to be entered or scanned. When the width is not specified, it will default to a value that allows the largest positive and the smallest negative value that the variable will accommodate to be input by the user. This specifier is exactly the same as the width specifier used in the ANSI C function scanf.

The optional precision specifier is a decimal integer that defines the maximum number of usable digits to the right of the decimal point. This specifier will be disregarded when the type specifier is in the 'diouxX' integer input set and shall be less than the width specifier (if present). When the precision specifier is absent from the edit format string, no data entry restriction (outside of the width specifier, if present) is placed on the digits entered to the right of the decimal point.

The type specifier is one of the 'diouxXeEfFgG' input types as specified for the scanf function.

9.1.15.4.4 SCALING_FACTOR

Purpose

The attribute 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. The SCALING_FACTOR shall not equal zero.

The SCALING_FACTOR can also be built by an expression (see Clause 18). The SCALING_FACTOR is applied for display purposes only. At all other times (e.g. when the VARIABLE is accessed from a method or compared to a minimum/maximum value) the value remains equivalent to and in the same form as the unscaled value returned by the device. Minimum and maximum values, when used in conjunction with the scaling factor, are also unscaled values.

SCALING_FACTOR shall not affect the DISPLAY_FORMAT of a VARIABLE.

Lexical structure

SCALING_FACTOR [*double-value*<cond> | *double-expression*<cond>];

9.1.15.4.5 size

Purpose

Size of the data type in octets. Size is an integer constant from 1 to 8 inclusive. This value is optional. The default is 1.

9.1.15.5 DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

9.1.15.5.1 General structure

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 milliseconds (ms) and a day count;
- the data type TIME consists of a time and a 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 unsigned integer of three octets Day, Month, Year. The coding of the octets is specified in B.2.2.

DATE

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

For the DEFAULT_VALUE, the constant-expression shall resolve to an unsigned 3-byte integer where the MSByte is Day and LSByte is (Year-1900). For example, 24 June 1993 could be expressed as:

```
DEFAULT_VALUE (( (24<<8)+6)<<8)+93;
```

Lexical structure – DATE_AND_TIME

DATE_AND_TIME specifies that VARIABLE is an unsigned integer of seven octets representing the calendar date in both time and date. The coding of the octets is specified in B.2.3.

DATE_AND_TIME

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

Lexical structure – DURATION

DURATION specifies that VARIABLE is an unsigned integer of 6 octets. The first four bits shall have the value zero. Bits 0 to 27 represent the time in milliseconds (ms) since midnight. The number of days is a 16-bit binary value. The coding of the octets is specified in B.2.4.

DURATION

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

Lexical structure – TIME

TIME specifies that VARIABLE is an unsigned integer of six octets. The time is stated in milliseconds (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 B.2.5.

TIME

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

Lexical structure – TIME_VALUE

TIME_VALUE specifies that VARIABLE is an unsigned 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 B.2.6.

TIME_VALUE (size)

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
  TIME_FORMAT string-value;
  TIME_SCALE time-scale-specifier;
}]?
```

For the DEFAULT_VALUE, the constant-expression shall resolve to an unsigned 4-byte or 8-byte integer. For example 4-byte TIME_VALUE encoding for 05:14:26 could be expressed as

DEFAULT_VALUE ((5*60+14)*60+26)*32000;

The MIN_VALUE and MAX_VALUE shall resolve to an unsigned 4-byte or 8-byte integer (like illustrated for DEFAULT_VALUE) or a reference to another VARIABLE of TYPE TIME_VALUE.

The attributes of the DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE types are specified in Table 25.

Table 25 – TYPE attributes for DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	m	x	m	x	x	m	m
9.1.15.4.2	DISPLAY_FORMAT	o	x	o	o	x	o	o
9.1.15.4.3	EDIT_FORMAT	o	x	o	o	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.2	MAX_VALUE	o	o	o	o	o	o	o
9.1.15.2	MIN_VALUE	o	o	o	o	o	o	o
9.1.15.5.4	size	o	x	o	x	x	o	o
9.1.15.5.2	TIME_FORMAT	o	o	o	o	o	o	o
9.1.15.5.3	TIME_SCALE	o	x	o	o	x	o	o

9.1.15.5.2 TIME_FORMAT

Purpose

The attribute 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 C programming language function strftime.

For DATE_AND_TIME, TIME and TIME_VALUE(8), the following additional time format specifiers apply: %a %A %b %B %c %d %D %e %F %g %G %h %m %x %y %Y.

When TIME_FORMAT is specified, DISPLAY_FORMAT and EDIT_FORMAT shall not be specified.

Lexical structure

TIME_FORMAT *string-value*<cond>

9.1.15.5.3 TIME_SCALE

Purpose

The attribute 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. Otherwise only the TIME_FORMAT attribute may be present.

If TIME_SCALE is absent then the default TIME_FORMAT shall be "%T" otherwise, the format shall default to the same as a VARIABLE of TYPE FLOAT.

TIME_SCALE is a scaling factor and rules of SCALING_FACTOR are applicable.

Lexical structure

TIME_SCALE [SECONDS , MINUTES , HOURS]<cond>

9.1.15.5.4 size

Purpose

Size of the data type in octets. Shall be either 4 or 8. For FF, the default value is 8 and for all other protocols, the value is 4. This attribute is only applicable to TIME_VALUE.

9.1.15.6 BIT_ENUMERATED

9.1.15.6.1 General structure

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.

An individual bit in a VARIABLE of type BIT_ENUMERATED may be referenced from elsewhere in the EDD via the variable name and the bit mask as specified in 16.3.

Like the value in the declaration, a bit-mask shall be constant-integer specifying a single bit.

Lexical structure

```

BIT_ENUMERATED (size)
{
    DEFAULT_VALUE unsigned-integer-value;
    INITIAL_VALUE unsigned-integer-constant;
    [{ value , description , help , class , status-class , action } , ]<cond>*
    [{ value , description , help , class , status-class , action } ]<cond>
}

```

The attributes of the BIT_ENUMERATED type are specified in Table 26.

Table 26 – TYPE attributes for BIT_ENUMERATED

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.15.6.4	action	o	o	o	o	o	o	o
9.1.2	class	o	o	o	o	o	o	o
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.6.2	description	m	m	m	m	m	m	m
14.8	help	o	o	o	o	o	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.6.5	size	o	o	o	o	o	o	o
9.1.15.6.6	status-class	o	o	o	o	o	o	o
9.1.15.6.3	value	m	m	m	m	m	m	m

9.1.15.6.2 description

Purpose

Description is the string that is displayed for this bit.

9.1.15.6.3 value

Purpose

Unsigned integer representing the significance of the bit. If the value of the VARIABLE supplies at the specified bit positions a logical one (1), the associated description is displayed. Equal values are not allowed.

9.1.15.6.4 action

Purpose

Reference to a METHOD that shall be executed when the bit is set. METHOD shall be executed when the bit becomes set (transitions from 0 to 1) during a read-operation or when the bit is read for the first time as a 1. Once the bit becomes set and the action is executed, the action will not run again until the bit transitions back to 0 and then to 1 again.

9.1.15.6.5 size

Purpose

Size of the data type in octets. Size is an integer constant from 1 to 8 inclusive. This value is optional. The default is 1.

Table 27 – size attribute

Value type	Description
unsigned-integer	Communication profiles: CS, GPE, HART, PB, PN Unsigned integer constant greater than zero.
unsigned-integer	Communication profiles: FF, ISA100 Unsigned integer constant greater than zero (limited to a maximum of 4).

9.1.15.6.6 status-class**Purpose**

Meaning of the bit. Zero or more status classes may be defined for a bit (see Table 28). Status classes are organized into categories, and only one status class from each category may be defined for a bit. For a given bit, zero or one value may be assigned from each category. When more than one status class is present, they are separated by the '&' symbol. If the VARIABLE is not a status octet, then the bits do not have status classes.

Table 28 shows the status-classes for a VARIABLE of type BIT ENUMERATED.

Table 28 – status-class attributes

Category	Attribute	Description
Cause:		
The cause status classes indicate to an EDD application why the bit was set. Zero or more of these status classes may be applied to an individual bit.	HARDWARE	A hardware failure.
	SOFTWARE	A software failure.
	PROCESS	A problem with the process connected to the device.
	MODE	The device is in a particular mode.
	DATA	An invalid data configuration.
	MISC	A miscellaneous condition.
Duration:		
The duration status classes indicates to an EDD application how long the bit may be set. Zero or one of these status classes may be applied to an individual bit.	EVENT	A one-time event.
	STATE	The device is in a particular state.
Correctability:		
The correctability status classes indicates to an EDD application whether the problem indicated by the bit being set can be corrected. Zero or one of these status classes may be applied to an individual bit.	SELF_CORRECTING	The bit will be reset without further intervention.
	CORRECTABLE	The bit can be cleared by the EDD application or the connected process.
	UNCORRECTABLE	The bit is neither self-correcting nor correctable.

Category	Attribute	Description
Scope:		
The scope status classes indicates whether the bit summarizes a problem in the device or provides additional detail to clarify a problem. Zero or one of these status classes may be applied to an individual bit.	SUMMARY	The bit is a logical combination of other bits of class detail.
	DETAIL	The bit is summarized elsewhere in a bit of class summary.
Miscellaneous:		
These status classes include those that do not fall into one of the other status classes. Zero or more of these status classes may be applied to an individual bit.	MORE	There is additional status information available from the device.
	COMM_ERROR	A communications failure in the other device.
	IGNORE_IN_TEMPORARY_MASTER	A bit should be ignored in temporary master devices.
Severity:		
The severity status classes indicate the importance or urgency an EDD application shall apply to the bit when set by the device. Only one of INFO, WARNING, or ERROR shall be applied to an individual bit. Any combination of IGNORE_IN_HOST or IGNORE_IN_HANDHELD may be associated with a bit.	INFO	The bit provides information about the status of the process or device.
	WARNING	The bit provides a warning about the status of the process or device.
	ERROR	The bit provides an error about the status of the process or device.
	IGNORE_IN_HOST	The bit shall be ignored in a fixed, permanently attached EDD application.
	IGNORE_IN_HANDHELD	A bit should be ignored in hand-held communicators.
Output status:		
	ALL	Impacts all the outputs (see Table 29).
	AO	Impacts one of the analog outputs (see Table 29).
	BAD	The output is unreliable and should not be used for control (see Table 29).
	DV	Impacts one of the dynamic variables (see Table 29).
	TV	Impacts one of the transmitter variables (see Table 29).

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

AOinteger ([AUTO & BAD | AUTO & GOOD | MANUAL & BAD | MANUAL & GOOD])

Lexical structure – DV

DVinteger ([AUTO & BAD | AUTO & GOOD | MANUAL & BAD | MANUAL & GOOD])

Lexical structure – TV

TVinteger ([AUTO & BAD | AUTO & GOOD | MANUAL & BAD | MANUAL & GOOD])

The attributes are specified in Table 29.

Table 29 – ALL, AO, DV, TV attributes

Attribute value	Description
AUTO & BAD	The value is coming from an application process and is invalid.
AUTO & GOOD	The value is coming from an application process and is valid.
integer	The VARIABLE value coming from the device.
MANUAL & BAD	The value is not coming from an application process and is invalid.
MANUAL & GOOD	The value is not coming from an application process and is valid.

Usage

An output class is the combination of either AUTO or MANUAL and either GOOD, MARGINAL or BAD. Therefore, the valid output classes are:

AUTO & GOOD AUTO & MARGINAL AUTO & BAD
MANUAL & GOOD MANUAL & MARGINAL MANUAL & BAD

If an output is AUTO, it is coming from the connected process; if an output is MANUAL, it is not coming from the connected process and is under the control of a human being. If an output is GOOD, the output is reliable and can be used for control; if an output is BAD, the output is unreliable and should not be used for control.

If the status bit is

DV1 (AUTO & BAD)

the primary variable is coming from the process and is unreliable.

If the status bit is

AO1 (MANUAL & GOOD)

the first analog output is not coming from the process and is reliable.

If the status bit is

TV6 (MANUAL & BAD)

the sixth device variable is not coming from the process and is unreliable.

If the status bit is

ALL (AUTO & BAD)

all the outputs of the device are coming from the process and are unreliable.

9.1.15.7 ENUMERATED

9.1.15.7.1 General structure

Purpose

A VARIABLE of type ENUMERATED is an unsigned integer defined as a set of values with associated text strings.

Lexical structure

```
ENUMERATED (size)
{
    DEFAULT_VALUE unsigned-integer-value;
    INITIAL_VALUE unsigned-integer-constant;
    [{ value , description , help } , ]<cond>*
    [{ value , description , help }]<cond>
}
```

The attributes of the ENUMERATED type are specified in Table 30.

Table 30 – TYPE attributes for ENUMERATED

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.3	description	m	m	m	m	m	m	m
9.1.15.3	help	o	o	o	o	o	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.7.2	size	o	o	o	o	o	o	o
9.1.15.3	value	m	m	m	m	m	m	m

9.1.15.7.2 size

Purpose

Size of the data type in octets. Size is an integer constant from 1 to 8 inclusive. This value is optional. The default is 1.

9.1.15.8 INDEX

Purpose

A VARIABLE of TYPE INDEX is an unsigned integer which is used to reference the elements of a VALUE_ARRAY, a REFERENCE_ARRAY (see 9.1) and LIST (see 9.1). A VARIABLE of TYPE INDEX may only take on the values defined by the REFERENCE_ARRAY, VALUE_ARRAY or LIST.

When the index is bound to a LIST the index variable shall automatically have a VALIDITY of FALSE when the list is empty.

When an INDEX VARIABLE is bound to a VALUE_ARRAY or LIST and presented to the user, the numeric value shall be displayed.

When an INDEX VARIABLE is bound to a REFERENCE_ARRAY and 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 [(size)]? [reference | reference<cond>;

The attributes of the index type are specified in Table 31.

Table 31 – TYPE attributes for INDEX

Value type	Description
size	Size of the VARIABLE in octets. Size of the data type in octets. Size is an integer constant from 1 to 4 inclusive. This value is optional. The default is 1. The REFERENCE_ARRAY or LIST shall contain only elements which do not exceed the index size of the variable. Communication profiles: HART An INDEX of size 1 shall only represent values of 0-249. The value of 250 is widely used in HART to indicate "Not Used" in indexed commands like Command 33 and 105. INDEX of size 2 has no similar restrictions.
reference	Reference to a REFERENCE_ARRAY, VALUE_ARRAY, or LIST instance.
reference<cond>	Communication profiles: CS, GRE, PB, PN Reference to a REFERENCE_ARRAY or LIST instance.

9.1.15.9 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, e.g. for a modular device (see IEC 61804-4).

Lexical structure

OBJECT_REFERENCE

9.1.15.10 ASCII, EUC, PACKED_ASCII, PASSWORD, VISIBLE

9.1.15.10.1 General structure

Purpose

String variable types include the following.

- ASCII – is for specifying a sequence of characters from the ISO Latin-1 character set.
- 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). EUC shall use the UTF-16 encoding form, limited to only characters from the Basic Multilingual Plane (BMP, Plane 0).
- PACKED_ASCII – is a subset of ASCII produced by removing the two most significant bits of each ASCII character (see B.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 646 character set number 2, plus SPACE character. It is restricted to 7-bit values in an 8-bit octet (high order bit is set to zero).

Lexical structure – EUC

```
EUC (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
}]?
```

Lexical structure – ASCII

```
ASCII (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
}]?
```

Lexical structure – PACKED_ASCII

```
PACKED_ASCII (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
}]?
```

Lexical structure – PASSWORD

```
PASSWORD (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
}]?
```

Lexical structure – VISIBLE

```
VISIBLE (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
}]?
```

The attributes of the string types are specified in Table 32.

Table 32 – TYPE attributes for ASCII, EUC, PACKED_ASCII, PASSWORD, VISIBLE

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.3	description	o	x	o	x	x	o	o
9.1.15.3	help	o	x	o	x	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.10.2	size	m	m	m	m	m	m	m
9.1.15.3	value	o	x	o	x	x	o	o

The EDD application shall display data types ASCII, EUC, PACKED_ASCII, PASSWORD and VISIBLE as strings. Termination or trimming of data types does not change the size of the data item. Termination and Trimming are used for string handling inside the EDD application only. Even if the data type can be terminated, the remaining characters of the data item will be filled by related fill characters. Depending on the data type the termination, filling characters and trailing characters shall be treated as shown in Table 33.

Table 33 – String handling

TYPE	Termination	Fill character	Trimming trailing spaces
ASCII	Null	Null	No
EUC	Null	Null	No
PACKED_ASCII	Not terminated	Space	No
PASSWORD	Communication profiles: HART, PB, PN: Null Communication profiles: FF, ISA100: not terminated	Communication profiles: HART, PB, PN: Null Communication profiles: FF, ISA100: space	No
VISIBLE	Not terminated	Space	Yes

9.1.15.10.2 size

Purpose

For the EUC string type, size specifies the number of octet pairs (16-bit words) reserved for the memory representation of the variable. Because UTF-16 is used and the characters are restricted to characters from BMP Plane 0, the number of characters that can be represented in the EUC string is the same as the value of the attribute size.

Except for the EUC string type, size specifies the number of characters from the appropriate character set, which may be different than the size of the memory representation of the variable.

Size is an integer constant greater than zero. This value is not optional; a size shall be specified.

9.1.15.11 OCTET

9.1.15.11.1 General structure

Purpose

OCTET is for specifying a sequence of unformatted binary data whose definition is determined by the implementation of the device.

Lexical structure – OCTET

OCTET (*size*)

```
[{
  DEFAULT_VALUE [{ integer-constant [, integer-constant]* } |
                   string-value ]<cond>;
  INITIAL_VALUE [{ integer-constant [, integer-constant]* } |
                   string-constant ];
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
}]?
```

The **DEFAULT_VALUE** and **INITIAL_VALUE** can take one of two forms:

- If the OCTET is intended to be used as a sequence of unformatted data bytes, the *integer-constants* shall resolve to a value from 0 to 255.
- If the OCTET is intended to be used as a sequence of characters from the ISO Latin-1 or UTF-8 character set, then the *string-value* or *string-constant* is used.

The attributes of the OCTET type are specified in Table 34.

Table 34 – TYPE attributes for OCTET

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.11.2	DISPLAY_FORMAT	o	o	o	x	o	o	o
9.1.15.11.3	EDIT_FORMAT	o	o	o	x	o	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.11.4	size	m	m	m	x	m	m	m

9.1.15.11.2 DISPLAY_FORMAT

Purpose

When specified, the **DISPLAY_FORMAT** specifies how the value of the VARIABLE is displayed.

Communication profiles: CS, GPE, PB, PN

When the **DISPLAY_FORMAT** attribute is not specified, a default format of "#x" shall be used.

Communication profiles: ISA100, FF

When the **DISPLAY_FORMAT** attribute is not specified, a default format of "X" shall be used.

A `DISPLAY_FORMAT` of "s" causes the `VARIABLE` of type `OCTET` to be interpreted as a sequence of characters from the ISO Latin-1 character set, and displayed as a string.

A `DISPLAY_FORMAT` of "x" or "X" causes the `VARIABLE` of type `OCTET` to be interpreted as a sequence of unformatted data bytes and displayed as a string of the values in hexadecimal format (e.g. "12 ab cd" etc.). Using lower case "x" shall cause the display of the values and prefix to use lower case letters. Alternatively, using upper case "X" shall display upper case letters. The leading flag # shall cause the hexadecimal prefix to be shown (#x for "0x", #X for "0X").

A `DISPLAY_FORMAT` of "o" causes the `VARIABLE` of type `OCTET` to be interpreted as a sequence of unformatted data bytes and displayed as a string of the values in octal format (e.g. "12 50 67", etc.). The leading flag # shall cause the octal prefix "0" to be shown.

A `DISPLAY_FORMAT` of "d" or "i" causes the `VARIABLE` of type `OCTET` to be interpreted as a sequence of unformatted data bytes and displayed as a string of the values in signed integer format (e.g. "-10 123 -40", etc.).

A `DISPLAY_FORMAT` of "u" causes the `VARIABLE` of type `OCTET` to be interpreted as a sequence of unformatted data bytes and displayed as a string of the values in unsigned integer format (e.g. "10 123 40", etc.).

A `DISPLAY_FORMAT` of "U" causes the `VARIABLE` of type `OCTET` to be interpreted as a sequence of characters from the UTF-8 (RFC 3629) character set. The usage of "U" is in addition to the display format used from ISO/IEC 9899.

Lexical structure

`DISPLAY_FORMAT` *string-value*<cond>;

9.1.15.11.3 EDIT_FORMAT

Purpose

Specifies the format for the editing. When an `EDIT_FORMAT` attribute is not specified, the `DISPLAY_FORMAT` used for the variable shall be used. The allowed edit formats are the same as those defined for the `DISPLAY_FORMAT` in 9.1.15.11.2.

Lexical structure

`EDIT_FORMAT` *string-value*<cond>;

9.1.15.11.4 size

Purpose

Size specifies the number of octets.

Size is an integer constant greater than zero. This value is not optional; a size shall be specified.

9.1.15.12 BITSTRING

9.1.15.12.1 General structure

Purpose

`BITSTRING` is for specifying a sequence of bits. When presented for displaying or editing, `BITSTRING` values may be represented either as a series of bits (1's and 0's), or as a sequence of octets represented in hexadecimal format.

Lexical structure – BITSTRING

BITSTRING (*size*)

```
[{
  DEFAULT_VALUE { integer-constant [, integer-constant]* }<cond>;
  INITIAL_VALUE { integer-constant [, integer-constant]* };
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
}]?
```

For DEFAULT_VALUE and INITIAL_VALUE, the *integer-constants* shall resolve to a value from 0 to 255, each value corresponding to an octet.

The attributes of the BITSTRING type are specified in Table 35.

Table 35 – TYPE attributes for BITSTRING

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.12.2	size	m	m	m	x	m	m	m

9.1.15.12.2 size

Purpose

Size specifies the number of bits, which shall be a multiple of 8.

Size is an integer constant greater than zero. This value is not optional; a size shall be specified.

9.1.15.13 BOOLEAN

Purpose

A VARIABLE of type BOOLEAN is a one byte unsigned integer with the value TRUE or FALSE. The value 0 reflects FALSE. Refer to Table B.12 for the value for TRUE.

Lexical structure

BOOLEAN

9.2 REFERENCE_ARRAY

9.2.1 General structure

Purpose

A REFERENCE_ARRAY is a set of items of the specified item type (for example, VARIABLE or MENU). Each item in the group is assigned a number, called an index.

REFERENCE_ARRAY has two available keywords, "ITEM_ARRAY" (first option) and "ARRAY" (second option) which may be selected by a profile. The keyword ITEM_ARRAY shall not be selected if the VALUE_ARRAY is used.

Lexical structure

Communication profiles: CS, GPE, FF, ISA100, PB, PN

reference-array-keyword := **ITEM_ARRAY**

Communication profiles: CS, GPE, HART, PB, PN

reference-array-keyword := **ARRAY**

reference-array-keyword **OF** *item-type identifier*

```
{
  LABEL string-value;
  HELP string-value;
  ELEMENTS { [[integer , reference [, description [, help]? ]?;<cond>]+ }
  PRIVATE boolean-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}
```

The attributes of REFERENCE_ARRAY are specified in Table 36.

Table 36 – REFERENCE_ARRAY attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		REFERENCE_ARRAY	✓	✓	✓	✓	✓	✓	✓
9.2.2		ELEMENTS	m	m	m	m	m	m	m
14.8		HELP	o	o	o	o	o	o	o
9.2.3		item-type	m	m	m	m	m	m	m
14.9		LABEL	o	o	o	o	o	o	o
14.13		PRIVATE	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o

Usage

The example in Figure 30 shows the definition of a REFERENCE_ARRAY named deviceVariables.


```

VARIABLE pressureValue
{
    CLASS CORRECTION & DYNAMIC;
    TYPE FLOAT;
}

VARIABLE pressureUnits
{
    TYPE ENUMERATED (2)
    {
        ALL_UNITS
    }
}

VARIABLE temperatureValue
{
    CLASS CORRECTION & DYNAMIC;
    TYPE FLOAT;
}

VARIABLE temperatureUnits
{
    TYPE ENUMERATED (2)
    {
        ALL_UNITS
    }
}

COLLECTION OF VARIABLE pressure
{
    MEMBERS
    {
        DIGITAL_VALUE,      pressureValue;
        DIGITAL_UNITS,      pressureUnits;
    }
}

COLLECTION OF VARIABLE temperature
{
    MEMBERS
    {
        DIGITAL_VALUE,      temperatureValue;
        DIGITAL_UNITS,      temperatureUnits;
    }
}

ARRAY OF COLLECTION deviceVariables
{
    ELEMENTS
    {
        0, pressure,
        1, temperature;
    }
}

```

Figure 30 – REFERENCE_ARRAY example

The example in Figure 31, shows the construction of an abstract command definition.

```

COMMAND read_dynamic_variables_and_pv_current
{
    REDEFINE TRANSACTION
    {
        REQUEST { }
        REPLY
        {
            response_code, device status,
            dynamic_variables[0].DAQ.ANALOG_VALUE,
            dynamic_variables[0].DEVICE_VARIABLE.DIGITAL_UNITS,
            dynamic_variables[0].DEVICE_VARIABLE.DIGITAL_VALUE
        }
    }
}

```

Figure 31 – Abstract COMMAND definition example

In this example, `dynamic_variables[0].DEVICE_VARIABLE.DIGITAL_VALUE` would reference `pressureValue`.

9.2.2 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. For description of help and label concatenation refer to IEC 61804-4.

Lexical structure

ELEMENTS { `[[integer , reference [, description [, help]? ]?;<cond>]+` }

The attributes of **ELEMENTS** are specified in Table 37.

Table 37 – ELEMENTS attribute

Value type	Description
integer	Number by which the item may be referenced. Each number shall be uniquely defined within a REFERENCE_ARRAY construct.
reference	Reference to an EDD item. Table 46 shows the allowed item-types.
description	Short description for the item.
help	Help text for the item.

9.2.3 item-type

Purpose

The **item-type** attribute specifies the type of members in the **REFERENCE_ARRAY**. All the **REFERENCE_ARRAY** members shall be of the specified type.

Lexical structure

item-type

Table 46 shows the allowed item-types.

9.3 VALUE_ARRAY

9.3.1 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.

Lexical structure

ARRAY *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  NUMBER_OF_ELEMENTS [integer-constant | unsigned-integer-constant |
                      integer-expression];
  PRIVATE boolean-specifier;
  RESPONSE_CODES reference;
  TYPE reference;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  WRITE_MODE write-mode-specifier;
}
```

The attributes of VALUE_ARRAY are specified in Table 38.

Table 38 – VALUE_ARRAY attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		VALUE_ARRAY	✓	✓	✓	✓	✓	✓	✓
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	m	m	m	m	m	m	m
9.3.2		NUMBER_OF_ELEMENTS	m	m	m	m	m	m	m
14.13		PRIVATE	o	o	o	o	o	o	o
14.15		RESPONSE_CODES	o	o	o	x	o	o	o
9.3.3		TYPE	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o
14.20		WRITE_MODE	x	o	x	x	o	x	x

9.3.2 NUMBER_OF_ELEMENTS

Purpose

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

Lexical structure

NUMBER_OF_ELEMENTS [integer-constant | unsigned-integer-constant | integer-expression];

The attributes of NUMBER_OF_ELEMENTS are specified in Table 39.

Table 39 – NUMBER_OF_ELEMENTS attribute

Value type	Description
integer-constant	Communication profiles: FF, ISA100 Number of elements of a VALUE_ARRAY.
unsigned-integer-constant	Communication profiles: HART Number of elements of a VALUE_ARRAY.
integer-expression	Communication profiles: CS, GPE, PB, PN Expression which shall be evaluated to a non-negative integer constant that is within the index range of the NUMBER_OF_ELEMENTS attribute; for the description of expressions, see Clause 18.

9.3.3 TYPE

Purpose

The TYPE attribute is a reference to an EDD basic construct instance. 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 40.

Table 40 – TYPE attribute

Attribute	Description
reference	Reference to an EDD basic construct instance. The reference shall terminate to a data item. The item referenced in the TYPE attribute of a VALUE_ARRAY shall not contain VALIDITY or any other conditional expressions that would cause data type structures to change. This restriction also applies for contained items. Communication profiles: FF, ISA100 Reference to a VARIABLE Communication profiles: CS, GPE, HART, PB, PN Reference to a VARIABLE, COLLECTION, LIST, VALUE_ARRAY or REFERENCE_ARRAY

Usage

For example, consider the following:

```
VARIABLE xValue { TYPE FLOAT; }
VARIABLE yValue { TYPE FLOAT; }
VARIABLE quality {
    TYPE enumerated { {0 , "Bad"}, {2 , "Poor"}, {3 , "Good"} }
}

COLLECTION point { MEMBERS { X, xValue; Y, yValue; QUALITY, quality; }

ARRAY dataSetArray { NUMBER_OF_ELEMENTS 100; TYPE point; }
```

This would create 100 collections of 3 variables. In other words, a total 300 variables would be created inside the array.

9.4 LIST

9.4.1 General structure

Purpose

A LIST is a variable length, insertable value-array. LIST describes a continuous sequence of elements where each element of the sequence is defined by an EDDL construct instance of a specific type.

Lexical structure

```
LIST identifier
{
  LABEL string-value;
  HELP string-value;
  CAPACITY [unsigned-integer-constant |
            unsigned-integer-expression<cond>];
  COUNT [unsigned-integer-constant | unsigned-integer-constant<cond> |
         unsigned-integer-expression<cond> | reference];
  PRIVATE boolean-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  TYPE reference;
}
```

The attributes of LIST are specified in Table 41.

Table 41 – LIST attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	LIST		✓	✓	✓	✓	✓	✓	✓
9.4.2		CAPACITY	o	o	o	o	o	o	o
9.4.3		COUNT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.13		PRIVATE	o	o	o	o	o	o	o
9.4.4		TYPE	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o

Usage

A list element is referenced from elsewhere in the EDD via the list identifier and the element index. This reference uses the square bracket notation. Due to the dynamic nature of lists, a list reference shall not be used in an expression. The syntax of this reference is described in 16.10.

Elements of a list are always contiguous and all list references shall be proper and well formed. Accessing a non-existent LIST element in a METHOD shall cause the method to abort. References to a non-existent LIST elements elsewhere in the EDD shall behave as if the referenced element has VALIDITY FALSE. List properties may also be referenced including COUNT, CAPACITY, FIRST, LAST.

```
list-identifier.COUNT
list-identifier.CAPACITY
list-identifier.FIRST
list-identifier.LAST
```

The COUNT property returns the number of elements currently residing in the lists. The CAPACITY indicates how many elements could be stored in the list.

Communication profiles: CS, GPE, HART, PB, PN

Using the property FIRST is equivalent to referencing list-identifier [0] and LAST is list-identifier[list-identifier.COUNT – 1].

Communication profiles: FF, ISA100

Using the property FIRST is equivalent to referencing list-identifier [1] and LAST is list-identifier[list-identifier.COUNT].

Lists are generally managed using methods and the corresponding builtins specified in IEC 61804-5. These builtins return error codes if an attempt to access a non-existent list element occurs.

9.4.2 CAPACITY

Purpose

The CAPACITY attribute specifies the maximum number of elements that may be stored in the list. When the CAPACITY attribute is not specified, it defaults to the maximum unsigned-integer value (+4 294 967 295) indicating that the LIST is not limited.

Lexical structure

CAPACITY [*unsigned-integer-constant* | *unsigned-integer-expression*<cond>];

The attribute of CAPACITY is specified in Table 42.

Table 42 – CAPACITY attribute

Value type	Description
unsigned-integer-constant	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Unsigned integer constant greater than or equal to one, and less than or equal to the maximum unsigned integer value (+4 294 967 295)
unsigned-integer-expression<cond>	Communication profiles: FF, ISA100 Unsigned integer expression, may be conditionally specified

9.4.3 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 as a constant, it defines the size that it has at creation time of the LIST instance. If it is not defined or defined as a reference, a LIST instance with zero elements is created.

Lexical structure

COUNT [*unsigned-integer-constant* | *unsigned-integer-constant*<cond> | *unsigned-integer-expression*<cond> | *reference*];

The attribute of COUNT is specified in Table 43.

Table 43 – COUNT attribute

Value type	Description
unsigned-integer-constant	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Unsigned integer constant
unsigned-integer-constant<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Unsigned integer constant, may be conditionally specified
unsigned-integer-expression<cond>	Communication profiles: FF, ISA100 Unsigned integer expression, may be conditionally specified
reference	Reference to a VARIABLE instance

The COUNT attribute, when referenced to a non-LOCAL variable, causes the list size to be controlled by the device; i.e. it is a device managed list.

The builtins ListInsert() and ListDeleteElementAt() cause the variable referenced by the COUNT attribute to be updated to match to the current number of elements in the LIST.

If the COUNT attribute resolves to an expression then none of the variables referenced in the expression will be updated as a result of the builtins.

COUNT shall not be allowed to be written in a METHOD.

Each time the variable referenced by the COUNT attribute changes, the whole LIST is required to be synchronized because the LIST structure is no longer valid.

In order to read a device managed LIST, the EDD application shall re-read all elements of the LIST from the device any time the value of the variable referenced by the COUNT attribute read from the device has changed.

In order to write a device managed LIST to the device, the EDD application shall write the variable referenced by the COUNT attribute and all elements of the LIST when the variable referenced by the COUNT attribute has been changed in the cache, e.g. by a METHOD. When the content of the LIST has been changed but no elements have been added or deleted, then only the changed LIST elements need to be written to the device.

9.4.4 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 44.

Table 44 – TYPE attribute

Value type	Description
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 reference shall terminate to a data item. 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.

9.5 COLLECTION

9.5.1 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 of the COLLECTION may be referenced from elsewhere in the EDD via the COLLECTION identifier and the MEMBER name using the dot notation:

collection-identifier.member-name

Each item name used within a COLLECTION shall be unique from all the other item names defined within the same COLLECTION.

COLLECTION is intended to be used, for example, to identify parameters that should be processed together.

Lexical structure

```
COLLECTION [OF item-type]? identifier
{
    LABEL string-value;
    HELP string-value;
    MEMBERS { members-specifier }
    PRIVATE boolean-specifier;
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
}
```

The attributes of COLLECTION are specified in Table 45.

Table 45 – COLLECTION attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		COLLECTION	✓	✓	✓	✓	✓	✓	✓
14.8		HELP	o	o	o	o	o	o	o
9.5.2		item-type	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
14.13		PRIVATE	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o

9.5.2 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.

Communication profiles: HART

Use is discouraged, supported for backward compatibility

Lexical structure

item-type

Table 46 shows the allowed item-types.

Table 46 – item-type

item-type	Description
BLOCK_A	Communication profiles: FF, ISA100 Selection of this basic construct.
BLOCK_B	Communication profiles: CS, GPE, PB, PN Selection of this basic construct.
CHART	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
COLLECTION	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
EDIT_DISPLAY	Communication profiles: FF, HART, ISA100 Selection of this basic construct.
FILE	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
GRAPH	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.

item-type	Description
GRID	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
IMAGE	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
LIST	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
MENU	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
METHOD	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
PLUGIN	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
RECORD	Communication profiles: FF, ISA100 Selection of this basic construct.
REFERENCE_ARRAY	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
VALUE_ARRAY	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
VARIABLE	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Selection of this basic construct.
VARIABLE_LIST	Communication profiles: FF, ISA100 Selection of this basic construct.

9.6 FILE

9.6.1 General structure

Purpose

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

A MEMBER may be a reference to a VARIABLE, LIST or VALUE_ARRAY. The FILE may only contain data and none of the data may be of class DYNAMIC.

NOTE If the MEMBER is not of CLASS LOCAL, then the data from the FILE will be overwritten when online with the device.

Lexical structure

```

FILE identifier
{
    LABEL string-value;
    HELP string-value;
    HANDLING handling-specifier;
    IDENTITY file-name;
    MEMBERS { members-specifier }
    ON_UPDATE_ACTIONS { [reference<cond>]+ }
    SHARED [TRUE | FALSE];
}

```

The attributes of FILE are specified in Table 47.

Table 47 – FILE attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	FILE		✓	✓	✓	✓	✓	✓	✓
14.6		HANDLING	o	x	o	o	x	o	o
14.8		HELP	o	o	o	o	o	o	o
9.6.2		IDENTITY	o	x	o	o	x	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
9.6.3		ON_UPDATE_ACTIONS	x	x	x	o	x	x	x
9.6.4		SHARED	o	x	o	o	x	o	o

9.6.2 IDENTITY

Purpose

The IDENTITY attribute designates the computer file or storage area for a given shared FILE and is used by EDD applications to identify it. This also means the external data set can be accessed by all device-type instances accordingly. It is up to the EDD application to access the correct FILE when the EDD application accesses an IDENTITY. The IDENTITY attribute is a quoted ASCII string of characters consisting of letters (a-z and A-Z), digits (0-9), the underscore character (_) or the period character (.). To ensure the IDENTITY of the FILE is not confused with that possibly used by other devices, EDD applications shall segregate their storage area for FILES by device type and, depending on the value of SHARED, possibly device instance.

If SHARED is specified and IDENTITY is not specified, then the identity of the FILE may be established using the browselidentity() builtin function. The IDENTITY attribute is directly accessible from within the EDD using the

```
file_reference.IDENTITY
```

string reference (i.e. it returns a character string matching the FILE's IDENTITY).

Lexical structure

```
IDENTITY file-name;
```

The attribute of IDENTITY is specified in Table 48.

Table 48 – IDENTITY attribute

Value type	Description
file-name	String literal that identifies the filename. Communication profiles: HART Length limited to 32 characters

9.6.3 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*<cond>]+ }

The attribute of ON_UPDATE_ACTIONS is specified in Table 49.

Table 49 – ON_UPDATE_ACTIONS attribute

Value type	Description
reference	Reference to a METHOD.

9.6.4 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.

Shared files are generally repositories of precalculated engineering data (e.g. tank geometries, flow tables).

NOTE Communication profiles: HART: Shared files are XML-based and conform to the XML schema controlled and provided by FieldComm Group. Contact FieldComm Group for more information.

Lexical structure

SHARED [**TRUE** | **FALSE**] ;

The attributes of SHARED are specified in Table 50.

Table 50 – SHARED attribute

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

9.7 Relations

9.7.1 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 and sent to the device. The modification may be initiated by user interactions inside the EDD application, inside METHOD executions or by reading a new value from the device. In order to prevent infinite execution of refresh relations, affected-parameters shall only be read if the cause-parameter has a value that is different from the value when it was last read. As a minimum, the refresh shall be done prior to using or displaying any of the dependent variables in the refresh relation.

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).

Communication profiles: FF, ISA100, CS, GPE, PB, PN

A VARIABLE may have a refresh relationship with itself, implying that the VARIABLE is to be read after writing. A dynamic value should not be placed in a relation with itself as this could cause unpredictable results.

Lexical structure

REFRESH identifier

```
{ [cause-parameter] | [cause-parameter<cond> [, cause-parameter]<cond>+] :  
  [affected-parameter [, affected-parameter]+] |  
  [affected-parameter<cond> [, affected-parameter<cond>+] ] )  
}
```

REFRESH is specified in Table 51.

Table 51 – REFRESH overview

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	REFRESH		✓	✓	✓	✓	✓	✓	✓

The attributes of REFRESH are specified in Table 52.

Table 52 – REFRESH attributes

Value type	Description
cause-parameter	Communication profiles: FF, ISA100 VARIABLE instance, RECORD or VALUE_ARRAY. Communication profiles: HART VARIABLE instance, LIST or VALUE_ARRAY.
cause-parameter<cond> [, cause-parameter]<cond>]+	Communication profiles: CS, GPE, PB, PN Set of VARIABLE instances, VALUE_ARRAYs. May be conditionally specified.
affected-parameter	Communication profiles: FF, ISA100 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. Communication profiles: HART Set of VARIABLE instances, LISTs 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.
affected-parameter<cond>	Communication profiles: CS, GPE, PB, PN Set of VARIABLE instances, LISTs, RECORDs or VALUE_ARRAYs, which shall be read if one of the cause parameters has been modified. May be conditionally specified. A VARIABLE of the affected parameter set may also be in the cause parameter set of another REFRESH relation.

9.7.2 SEMANTIC_MAP

Purpose

SEMANTIC_MAP relations provide a generic access to data items via standardized semantic identifiers, without knowing any internal EDD item names. Semantic identifiers are locale-independent strings typically specified in international standards like the IEC Common Data Dictionary (IEC 61987). In order to avoid conflict with various name spaces in these identifiers, the structure of the string used shall be based on ISO/TS 29002-5.

One example for a semantic identifier is "0112/2///61987#ABB271#002", the semantic identifier of the "tag name/PCE request name" property.

A SEMANTIC_MAP relation defines a "key/value" pair where "key" contains the semantic identifier and "value" the reference to the related data item.

If the referenced data item is an enumeration VARIABLE, the reference to the data item can be extended with an enumeration list for mapping the enumeration values to semantic identifiers.

If the unit of the referenced data item is specified as CONSTANT_UNIT, the reference to the data item can be extended by adding the attribute CONSTANT_UNIT and the respective semantic identifier for the unit.

Lexical structure

SEMANTIC_MAP *identifier*

```
{
  [semantic-id] | [semantic-id [, semantic-id]+]:
  [semantic-item [, semantic-item]+]
}
semantic-item:= reference [{{{ value , semantic-id-enum } , }*
                        [{ value , semantic-id-enum }]]?
                        [CONSTANT_UNIT semantic-id-unit;]? }]]?
```

SEMANTIC_MAP is specified in Table 53.

Table 53 – SEMANTIC_MAP overview

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	SEMANTIC_MAP		✓	✓	✓	✓	✓	✓	✓

The attributes of SEMANTIC_MAP are specified in Table 54.

Table 54 – SEMANTIC_MAP attributes

Value type	Description
semantic-id	String literal specified as semantic identifier
semantic-item	Reference to the related data item
semantic-id-enum	String literal specified as semantic identifier for an enumeration value
semantic-id-unit	String literal specified as semantic identifier for the unit

Usage

Figure 32 shows several SEMANTIC_MAP definitions.

```
// --- Example 1 ---
SEMANTIC_MAP map_tag
{
    "0112/2///61987#ABB271#002":
    var_tag
}

// --- Example 2 ---
SEMANTIC_MAP map_damping
{
    "<semID_damp_A>", "<semID_damp_B>", "<semID_damp_C>":
    var_damping
    {
        CONSTANT_UNIT "<semID_seconds>";
    }
}

// --- Example 3 ---
SEMANTIC_MAP map_meas_type
{
    "<semID_meas_type>":
    var_meas_type
    {
        { 2, "<semID_meas_type_absPress>" }, // Measuring type: differential pressure
        { 3, "<semID_meas_type_diffPress>" } // Measuring type: absolute pressure
    }
}
```

Figure 32 – EDD examples of SEMANTIC_MAP

Example 1:

shows a 1:1 relation to map the semantic identifier "0112/2///61987#ABB271#002", the semantic identifier of the "tag name/PCE request name" property, to the VARIABLE holding the device tag.

Example 2:

shows an n:1 relation to map multiple semantic identifiers, in this example 3 identifiers, to one VARIABLE.

Since the unit of VARIABLE "var_damping" is specified as CONSTANT_UNIT, the reference to "var_damping" has been extended with the CONSTANT_UNIT attribute and the semantic identifier for unit "seconds".

Example 3:

shows a relation to map a semantic identifier to an enumeration VARIABLE. Further the enumeration list values are mapped to the respective semantic identifiers.

9.7.3 UNIT

Purpose

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

Lexical structure

UNIT identifier

```
{ [cause-parameter | cause-parameter<cond>]:
    [affected-parameter [, affected-parameter]+] |
    [affected-parameter<cond> [, affected-parameter<cond>]+])
}
```

UNIT is specified in Table 55.

Table 55 – UNIT overview

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	UNIT		✓	✓	✓	✓	✓	✓	✓

The attributes of UNIT are specified in Table 56.

Table 56 – UNIT attributes

Value type	Description
cause-parameter	Communication profiles: FF, HART, ISA100 Reference to an enumerated type VARIABLE instance containing the unit code or to a string type VARIABLE instance containing the unit string.
cause-parameter<cond>	Communication profiles: CS, GPE, PB, PN Reference to an enumerated type VARIABLE instance containing the unit code or to a string type VARIABLE instance containing the unit string. May be conditionally specified.
affected-parameter	Communication profiles: FF, ISA100 List of VARIABLE, VALUE_ARRAY, LIST or AXIS instances using that unit code. Communication profiles: HART List of VARIABLE or AXIS instances using that unit code
affected-parameter<cond>	Communication profiles: CS, GPE, PB, PN List of VARIABLE, VALUE_ARRAY, LIST or AXIS instances using that unit code. May be conditionally specified.

9.7.4 WRITE_AS_ONE

Purpose

A WRITE_AS_ONE relation specifies a list of data items that shall be modified together. The EDD application shall display all elements of the relation whenever any one of these items is being edited, 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 should not be used.

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 [, reference]]+ | [reference<cond> [, reference<cond>]]+
}
```

WRITE_AS_ONE is specified in Table 57.

Table 57 – WRITE_AS_ONE overview

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	WRITE_AS_ONE		✓	✓	✓	✓	✓	✓	✓

The attribute of WRITE_AS_ONE is specified in Table 58.

Table 58 – WRITE_AS_ONE attribute

Attribute	Description
reference	Communication profiles: FF, ISA100 List of VARIABLES, VALUE_ARRAYs and RECORDs instances that shall be displayed. Communication profiles: HART List of VARIABLES that are not BIT_ENUMERATED shall be displayed. VALUE_ARRAYs and LIST elements of TYPE VARIABLE that are not BIT_ENUMERATED shall be displayed.
reference<cond>	Communication profiles: CS, GPE, PB, PN List of VARIABLES and VALUE_ARRAYs instances that shall be displayed.

9.8 BLOB

9.8.1 General structure

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, e.g. for validation purposes only. A BLOB is a data container whose content is accessible via builtins. An EDD application may restrict the size of a BLOB but shall allow transfers of at least 1 048 575 bytes of data.

The HANDLING attribute shall either READ or WRITE (not both).

NOTE HANDLING READ means the BLOB can only be read from the device.

The IDENTITY attribute is directly accessible from within the EDD using the

`blob_reference.IDENTITY`

string reference (i.e. it returns a character string matching the BLOB's IDENTITY).

The size of the BLOB is directly accessible from within the DD using the

`blob_reference.COUNT`

integer reference that returns the size of the BLOB (i.e. the number of bytes).

Lexical structure

BLOB *identifier*

```
{
    LABEL string-value;
    HELP string-value;
    HANDLING handling-specifier;
    IDENTITY file-name;
}
```

The attributes of BLOB are specified in Table 59.

Table 59 – BLOB attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		BLOB	x	x	x	✓	x	x	x
14.6		HANDLING	x	x	x	m	x	x	x
14.8		HELP	x	x	x	o	x	x	x
9.8.2		IDENTITY	x	x	x	o	x	x	x
14.9		LABEL	x	x	x	o	x	x	x

9.8.2 IDENTITY

Purpose

The IDENTITY (i.e. the BLOB's filename) designates the computer file or storage area for a given BLOB and is used by applications to identify the BLOB. The IDENTITY attribute is a quoted ASCII string of up to 32 characters long consisting of letters (a-z and A-Z), digits (0-9), the underscore character (_) or the period character (.). To ensure the IDENTITY of the BLOB is not confused with that possibly used by other devices, EDD applications shall segregate their storage area for BLOBs by device type.

If IDENTITY is not defined, the filename shall be selected using a method (see IEC 61804-5).

Lexical structure

IDENTITY *file-name*;

The attribute of IDENTITY is specified in Table 60.

Table 60 – IDENTITY attribute

Value type	Description
file-name	String literal that identifies the filename. Communication profiles: HART Length limited to 32 characters

10 Communication

10.1 COMMAND

10.1.1 General structure

Purpose

COMMAND is a construct to support mapping of the EDD communication elements to a chosen communication system. A COMMAND specifies the data being transferred from and to a device. The data to be written to the device is specified in the REQUEST attribute. The data to be read from the device is specified in the REPLY attribute. It specifies various elements required to build a communication frame.

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*

```
{
  API [integer-constant | reference | integer-expression]<cond>;
  BLOCK_B reference<cond>;
  HEADER string-value<cond>
  INDEX [integer-constant | reference | integer-expression]<cond>;
  NUMBER integer-constant;
  OPERATION [COMMAND | READ | WRITE];
  RESPONSE_CODES { [referenced-response-codes-specifier |
                    response-codes-specifier] }
  SLOT [integer-constant | reference | integer-expression]<cond>;
  SUB_SLOT [integer-constant | reference | integer-expression]<cond>;
  TRANSACTION [integer]?
  {
    REQUEST { request-item-specifier [, request-item-specifier]* }
    REPLY { reply-item-specifier [, reply-item-specifier]* }
    RESPONSE_CODES { [referenced-response-codes-specifier |
                     response-codes-specifier] }
    POST_REQUEST_RECEIVE_ACTIONS { actions-specifier }
  }
}
```

The attributes of COMMAND are specified in Table 61.

Table 61 – COMMAND attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	COMMAND		✓	x	✓	✓	x	✓	✓
10.1.3	API		o	x	o	x	x	x	o
10.1.4	BLOCK_B		o	x	x	x	x	o	o
10.1.5	HEADER		o	x	o	x	x	x	x
10.1.6	INDEX		o	x	x	x	x	m	m
10.1.7	NUMBER		o	x	x	m	x	x	x
10.1.8	OPERATION		m	x	m	m	x	m	m
14.15	RESPONSE_CODES	Communication profiles: HART Only: embedded RESPONSE_CODES; no conditionals. COMMAND RESPONSE_CODES are mandatory if there are no RESPONSE_CODES in the TRANSACTION	o	x	o	c	x	o	o
10.1.9	SLOT		o	x	x	x	x	o	o
10.1.10	SUB_SLOT		o	x	x	x	x	x	o
10.1.11	TRANSACTION	Communication profiles: CS, GPE, PB, PN Only one TRANSACTION with no integer definition	m	x	m	m	x	m	m

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.

10.1.2 Command addressing

10.1.2.1 HART command addressing

The NUMBER specifies the HART command number that the device use to identify the command and the data.

10.1.2.2 PROFIBUS DP addressing

SLOT and INDEX specify the PROFIBUS dataset that is transferred. In a modular device the modules are typically addressed with the SLOT and the data of a module is addressed with the INDEX.

10.1.2.3 PI Profile for Process Control Devices addressing

The absolute addresses of BLOCKs of a PI Profile for Process Control Devices device are stored in the device management directory. The EDD application needs to fetch the directory and to resolve the base address of the BLOCK (BLOCK_B). The BLOCK attribute of a COMMAND is a reference to a BLOCK_B item.

A BLOCK_B EDDL item is described with a TYPE and a NUMBER attribute. The TYPE attribute defines the types of a block (PHYSICAL, TRANSDUCER or FUNCTION). The NUMBER attribute defines the number of a block of a block type in the device management directory. NUMBER starts for all block types with 1.

The command attribute INDEX is a relative index to the first dataset of the block. To address a dataset in the device, the absolute starting slot and starting index from the device management directory are used and the relative INDEX of the COMMAND is added.

10.1.2.4 PROFINET addressing

API, SLOT, SUB_SLOT and INDEX are used to address datasets in the device in an absolute way.

10.1.2.5 Addressing for other protocols

HEADER is a string attribute that allows to contain protocol specific information for a COMMAND. The HEADER attribute is not used for PROFIBUS, PROFINET, FOUNDATION Fieldbus, ISA100_Wireless or HART devices. The concrete contents of the string is out of the scope of this document.

10.1.3 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-constant* | *reference* | *integer-expression*]*<cond>*;

The attributes of API are specified in Table 62.

Table 62 – API attribute

Attribute	Description
integer-constant	Number of the API.
reference	Reference to a VARIABLE instance containing the API number.
integer-expression	Expression, which shall be evaluated to a non-negative integer for the API number; for the description of expressions, see Clause 18.

10.1.4 BLOCK_B

Purpose

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

Lexical structure

BLOCK_B *reference**<cond>*;

The attribute of BLOCK_B is specified in Table 63.

Table 63 – BLOCK_B attribute

Value type	Description
reference	Reference to a BLOCK_B instance.

10.1.5 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-value*<cond>

The attribute of HEADER is specified in Table 64.

Table 64 – HEADER attribute

Attribute	Description
string-value	String that is used for the communication with specific communication protocols.

10.1.6 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-constant* | *reference* | *integer-expression*]<cond>;

The attributes of INDEX are specified in Table 65.

Table 65 – INDEX attribute

Value type	Description
integer-constant	Number of the index.
reference	Reference to a VARIABLE instance containing the index.
integer-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 Clause 18.

10.1.7 NUMBER

Purpose

The NUMBER attribute is used to specify the COMMAND number.

Lexical structure

NUMBER *unsigned-integer-constant*;

The attribute of NUMBER is specified in Table 66.

Table 66 – NUMBER attribute

Attribute	Description
unsigned-integer-constant	HART command number.

10.1.8 OPERATION

Purpose

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

OPERATION READ specifies that the primary purpose of this COMMAND is to read data items from the device. But, depending on the protocol or device specific addressing mechanisms a READ command can specify data to be transferred to the device, e.g. INFO, INDEX variables before data is received from the device. With this data the device can address the data items that should be transferred to the EDD application.

OPERATION WRITE specifies that the COMMAND writes data items to the device. A WRITE command can specify data in the REPLY attribute that is transferred from the device after writing. If the REPLY attribute contains data items that are written to the device with this command, data modifications made by the device can be detected in the EDD application.

OPERATION COMMAND specifies that the COMMAND purpose is to trigger a function. REQUEST specifies the input parameters. REPLY specifies the output parameter. The EDD application shall not use such COMMANDs for data transfer. Such COMMANDs may only be used by using communication builtins in METHODS.

Lexical structure

OPERATION [**COMMAND** | **READ** | **WRITE**];

The attributes of OPERATION are specified in Table 67.

Table 67 – OPERATION attribute

Attribute value	Description
COMMAND	The device performs a predefined set of actions, e.g. self-test, master reset.
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.
WRITE	Specifies a write transaction. The device receives the current values of the specified variables and sets the values of the specified set of variables. VARIABLES shall be defined in the REQUEST section of TRANSACTION.

Table 68 defines for PROFIBUS and PROFINET the communication mapping to the communication services.

Table 68 – PROFIBUS and PROFINET communication mapping

REQUEST items	REPLY items	Command
no	no	EDD application shall use a write service
yes	no	EDD application shall use a write service
no	yes	EDD application shall use a read service
yes	yes	Is not allowed. The EDD application shall generate an error and not perform any communication.

10.1.9 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-constant* | *reference* | *integer-expression*] <cond>;

The attributes of SLOT are specified in Table 69.

Table 69 – SLOT attribute

Value type	Description
integer-constant	Number of the slot.
reference	Reference to a VARIABLE instance containing the slot number.
integer-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 Clause 18.

10.1.10 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-constant* | *reference* | *integer-expression*] <cond>;

The attributes of SUB_SLOT are specified in Table 70.

Table 70 – SUB_SLOT attribute

Value type	Description
integer-constant	Number of the sub-slot.
reference	Reference to a VARIABLE instance containing the sub-slot number.
integer-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 Clause 18.

10.1.11 TRANSACTION

10.1.11.1 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 for communication profile HART. When a command has multiple transactions, each transaction shall be marked with a number after the TRANSACTION keywords so that the transactions may be distinguished from one another. When the transaction number is not specified it will be assumed to be zero. Each TRANSACTION may include a set of response codes that apply only to that particular transaction.

Lexical structure

TRANSACTION [*integer*]?

```
{
  REQUEST { request-item-specifier [, request-item-specifier]* }
  REPLY { reply-item-specifier [, reply-item-specifier]* }
  RESPONSE_CODES { [referenced-response-codes-specifier |
                      response-codes-specifier] }
}
```

The attributes of TRANSACTION are specified in Table 71.

Table 71 – TRANSACTION attributes

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
	integer	o	x	o	o	x	o	o
10.1.11.2	REPLY Communication profiles: HART No conditionals. REPLY may only contain VARIABLE references; complete instances of LIST, VALUE_ARRAY, REFERENCE_ARRAY, COLLECTION are not supported.	m	x	m	m	x	m	m
10.1.11.3	REQUEST Communication profiles: HART No conditionals. REQUEST may only contain VARIABLE references; complete instances of LIST, VALUE_ARRAY, REFERENCE_ARRAY, COLLECTION are not supported.	m	x	m	m	x	m	m
14.15	RESPONSE_CODES Communication profiles: HART Only: embedded RESPONSE_CODES; no conditionals. TRANSACTION RESPONSE_CODES are mandatory if there are no RESPONSE_CODES in the COMMAND	o	x	o	c	x	o	o
10.1.11.6	POST_RQSTRECEIVE_ACTIONS	x	x	x	o	x	x	x

10.1.11.2 REPLY

Purpose

The REPLY attribute contains a list of data items, which are received from the device. The order of the list of data items 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, VALUE_ARRAY, REFERENCE_ARRAY, and COLLECTION with an item-mask is not allowed.

When a data item does not begin and end on octet boundaries, a item-mask should be used to specify how the variable is extracted from the data field. 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.

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

Lexical structure

```
REPLY { reply-item-specifier [, reply-item-specifier]* }
reply-item-specifier:= [integer-constant | [reference [< item-mask >]]?
[INDEX | INFO][?]]
```

The attributes of REPLY are specified in Table 72.

Table 72 – REPLY and REQUEST attributes

Attribute	Description
integer	Integer literal which is always one-octet in size.
reference	Reference to a VARIABLE instance Communication profiles: PB, PN, CS, GPE Additionally, reference to a VALUE_ARRAY, COLLECTION, REFERENCE_ARRAY, or LIST instance.
item-mask	Bit pattern to extract the VARIABLE from the data field (see 10.1.11.4). 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 correspond to the length of the data item in the REQUEST or REPLY list. Communication profiles: HART No gaps between item masks shall be used.
INFO	The VARIABLE is not actually stored in the device or EDD application. It provides additional information to ensure correct processing.
INDEX	The VARIABLE is used in the REQUEST or REPLY as an index into a REFERENCE_ARRAY, LIST and VALUE_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 or EDD application.

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

10.1.11.3 REQUEST

Purpose

The REQUEST attribute contains a list of data items, which are sent to the device. The order of the list of data items 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, VALUE_ARRAY, REFERENCE_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.

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 { request-item-specifier [, request-item-specifier]* }
request-item-specifier:= [integer-constant | [reference [< item-mask >]?
[INDEX | INFO]?]]
```

The attributes of REQUEST are specified in Table 72.

10.1.11.4 item-mask

Purpose

Item masks are only needed when VARIABLES or constants do not begin and end on byte boundaries. An item without an item mask starts at bit 0. Item masks may be specified for VARIABLES and constants. The bits with the value '1' in the item mask define the mapping of the data item to the data frame. The bits with the value '1' shall be consecutive. If the value of bit 0 of the item-mask is '1', the following item is mapped to bit 0 of the next byte in the data frame. Bits not contained in the item mask are set to 0. Overlapping between item masks is only allowed if bit 0 of the previous mask is set.

In a reply packet,

- the command bit stream is masked (bit-wise and) with the item mask value;
- shifted right by the number of consecutive trailing '0' bits in the item mask and then;
- the corresponding bits in the data-item are set to the resultant value; and
- the other bits in the destination variable are unaffected.

In a request packet,

- the data-item is shifted left by the number of consecutive trailing '0' bits in the item mask;
- the result is masked (bit-wise and) with the item mask value; and
- that result is inserted into the corresponding bits in the currently active byte(s) in the packet.

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.

Multi-byte item masks define a mapping to a larger range in the data frame. The number of bits with the value '1' in the item mask shall be less than or equal to the number of bits of the data item.

If an item mask is specified for a VARIABLE or constant and the previous and next item do not have an item mask, then the first bit set in the mask defines the start position and the last bit set the end position in the data frame.

The masks shall be specified from MSB to LSB in the data frame.

Communication profiles: HART

Gaps in the item mask are not allowed.

Usage

Figure 33 illustrates the use of data item masks.

```
REQUEST {
  x <0xF8>, y <0x07>
}
```

Figure 33 – Example of an item mask

The data field of the request contains two VARIABLES which are packed into one byte. The VARIABLE x is packed into the five most significant bits and y is packed into the three least significant bits. If x was equal to 155 and y was equal to 5, the transmitted byte would be 221 (0xDD).

```
COMMAND cmd_abc
{
  OPERATION WRITE; // or READ
  ...             // protocol specific attributes, e.g. SLOT and INDEX
  TRANSACTION
  {
    REQUEST // or REPLY in case of READ
    {
      a,
      b <0x3c>, // 00111100b
      c
    }
  }
}
```

Figure 34 – Example of a single item mask

In Figure 34 a, b and c are INTEGER(1) VARIABLES. In this case bit 0 to bit 3 of VARIABLE b are mapped to bit 2 to bit 5 in the data frame behind VARIABLE a. VARIABLE c starts on bit 0 of the next byte in the data frame (see Figure 35).

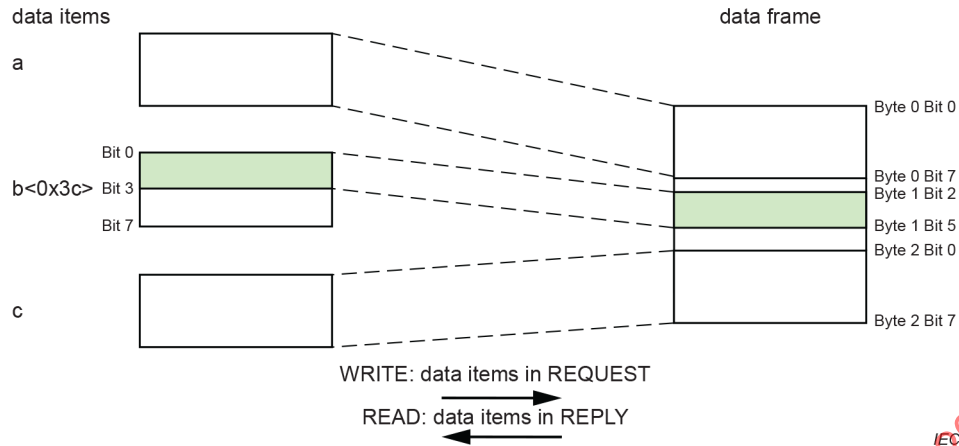


Figure 35 – Mapping example with a single item mask

In Figure 36 a, c, d, and e are INTEGER(1) VARIABLES, and b is an INTEGER(2) VARIABLE. In this case Bit 0 to Bit 11 of VARIABLE b are mapped to Bit 0 of Byte 1 to Bit 3 of Byte 2 in the data frame. Bit 0 to Bit 1 of VARIABLE c are mapped to Bit 4 to Bit 5 of Byte 2 in the data frame. Bit 0 of VARIABLE d is mapped to Bit 7 of Byte 2 (see Figure 37).

```

COMMAND cmd_abcde
{
  OPERATION WRITE; // or READ
  ... // protocol specific attributes, e.g. SLOT and INDEX
  TRANSACTION
  {
    REQUEST // or REPLY in case of READ
    {
      a,
      b <0xFFFF>, // 1111111111110000b
      c <0x000C>, // 000000000001100b
      d <0x0002>, // 000000000000010b, required only in HART, no gaps allowed
      e <0x0001>, // 0000000000000001b
    }
  }
}
    
```

Figure 36 – Example of multiple item masks

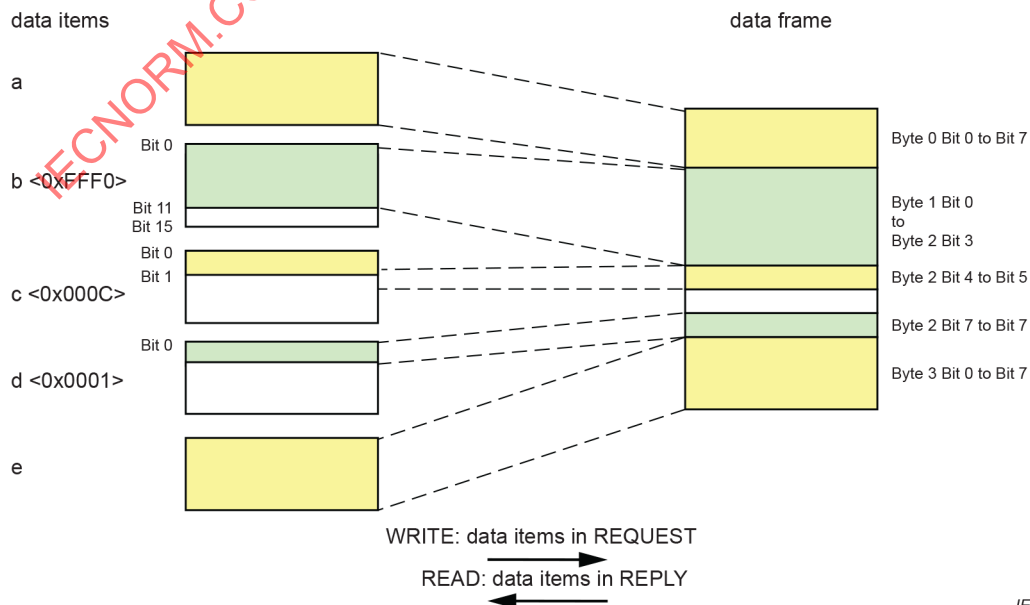


Figure 37 – Mapping example with a multiple item mask

10.1.11.5 Data item qualifiers

Purpose

A VARIABLE appearing in a request or reply may be qualified with the keywords INDEX and INFO.

A VARIABLE qualified with INDEX specifies that the VARIABLE is used in the request or reply as array index.

A VARIABLE qualified with INFO specifies the variable is not actually stored in the device or EDD application, the VARIABLE is only communicated for informational purposes.

A VARIABLE may be qualified with both INDEX and INFO; in this case, the VARIABLE is used as array index and is not actually stored in the device.

Usage

Figure 38 shows an EDD example demonstrating the use of an INFO qualified VARIABLE.

```
TRANSACTION
{
  REQUEST
  {
    units (INFO), x, y
  }
  REPLY
  {
    units, x, y
  }
}
```

Figure 38 – INFO qualifier

The VARIABLES *x* and *y* are written to the device in the unit which is specified by 'units'. The VARIABLES *x* and *y* are stored in the device (presumably in a fixed unit), but 'units' is not. This allows a VARIABLE to be communicated in a unit different from its current unit (the unit in which the variable is displayed).

Figure 39 shows an EDD example demonstrating the use of an INDEX qualified VARIABLE.

```
TRANSACTION
{
  REQUEST
  {
    code (INDEX), table[code]
  }
  REPLY
  {
    code, table[code]
  }
}
```

Figure 39 – INDEX qualifier

The data item in the array named *table* specified by a *code* is written to the device. The device stores the value of the *code* and echoes its value. The data item sent along with the *code* is also stored and the value stored in the device is returned.

Figure 40 shows an example demonstrating the use of local index VARIABLES.

```

TRANSACTION
{
    REQUEST
    {
        code (INFO, INDEX), table[code]
    }
    REPLY
    {
        code, table[code]
    }
}
    
```

Figure 40 – INFO and INDEX qualifier

The VARIABLE 'code' with the INFO and INDEX qualifier is sent to the device, but not stored in the device or EDD application. The VARIABLE is only used as index for the array 'table'.

10.1.11.6 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<cond> [, reference<cond>]*
}
    
```

The attribute of POST_RQSTRECEIVE_ACTIONS is specified in Table 73.

Table 73 – POST_RQSTRECEIVE_ACTIONS attributes

Value type	Description
reference	Reference to a METHOD.

10.2 RECORD

Purpose

A RECORD is a logical group of VARIABLES used to represent a communication object. Each MEMBER in the RECORD is a reference to a VARIABLE and may be of different data types. The VARIABLES may be referenced from elsewhere in the EDD via the RECORD identifier and the MEMBER identifier.

Lexical structure

RECORD *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  MEMBERS { members-specifier }
  PRIVATE boolean-specifier;
  RESPONSE_CODES referenced-response-codes-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  WRITE_MODE write-mode-specifier;
}
```

The attributes of RECORD are specified in Table 74.

Table 74 – RECORD attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	RECORD		x	✓	x	x	✓	x	x
14.8	HELP		x	o	x	x	o	x	x
14.9	LABEL		x	m	x	x	m	x	x
14.12	MEMBERS		x	m	x	x	m	x	x
14.13	PRIVATE		x	o	x	x	o	x	x
14.15	RESPONSE_CODES		x	o	x	x	o	x	x
		Communication profiles: FF, ISA100							
		Only referenced RESPONSE_CODES.							
14.17	VALIDITY		x	o	x	x	o	x	x
14.18	VISIBILITY		x	o	x	x	o	x	x
14.20	WRITE_MODE		x	o	x	x	o	x	x

10.3 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 , response-code-type , description , [help]?]<cond>;]+
}
response-code-type:= [DATA_ENTRY_ERROR | DATA_ENTRY_WARNING | MISC_ERROR |
MISC_WARNING | MODE_ERROR | PROCESS_ERROR | SUCCESS]
```

RESPONSE_CODES is specified in Table 75

Table 75 – RESPONSE_CODES overview

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	RESPONSE_CODES		✓	✓	✓	x	✓	✓	✓

The attributes of RESPONSE_CODES are specified in Table 76.

Table 76 – RESPONSE_CODES attributes

Attribute	Description
description	String that will be displayed when the response code is returned by the device.
integer	The returned value of the device to identify the response code.
DATA_ENTRY_ERROR	The application layer service was rejected because the data sent was improper.
DATA_ENTRY_WARNING	The application layer service was accepted and processed with a slightly modified version of the data sent.
MISC_ERROR	The application layer service was rejected.
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.
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.
PROCESS_ERROR	The application layer service was rejected because a process applied to the device was improper.
SUCCESS	The application layer service was accepted and processed as specified.
help	Help text for the item.

10.4 VARIABLE_LIST

Purpose

A VARIABLE_LIST is a group of EDD communication objects (VARIABLE, VALUE_ARRAY or RECORDS). The communication objects may be referenced from elsewhere in the EDD via the VARIABLE_LIST identifier and the MEMBER identifier. VARIABLE_LISTs are used to group objects for application convenience. VARIABLE_LISTs allow optimization in communication by offering the opportunity to group multiple items into fewer communication objects.

Lexical structure

VARIABLE_LIST identifier

```
{
  LABEL string-value;
  HELP string-value;
  MEMBERS { members-specifier }
  RESPONSE_CODES referenced-response-codes-specifier;
}
```

The attributes of VARIABLE_LIST are specified in Table 77.

Table 77 – VARIABLE_LIST attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		VARIABLE_LIST	x	✓	x	x	✓	x	x
14.8		HELP	x	o	x	x	o	x	x
14.9		LABEL	x	o	x	x	o	x	x
14.12		MEMBERS	x	m	x	x	m	x	x
14.15		RESPONSE_CODES	x	o	x	x	o	x	x

11 User interface

11.1 MENU

11.1.1 General structure

Purpose

MENU organizes variables, methods, and other items specified in the EDDL into a hierarchical structure. MENUs are the primary mechanism to organize device information and support a user's interaction with the device. A user application may use the MENU ITEMS to display and enter information in an organized and consistent fashion. MENU hierarchies are created by defining a menu, and then including it as an item of another MENU.

Lexical structure

MENU *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  ACCESS [OFFLINE | ONLINE];
  EXIT_ACTIONS { actions-specifier }
  INIT_ACTIONS { actions-specifier }
  ITEMS { menu-item-specifier [, menu-item-specifier]* }
  POST_EDIT_ACTIONS { actions-specifier }
  POST_READ_ACTIONS { actions-specifier }
  POST_WRITE_ACTIONS { actions-specifier }
  PRE_EDIT_ACTIONS { actions-specifier }
  PRE_READ_ACTIONS { actions-specifier }
  PRE_WRITE_ACTIONS { actions-specifier }
  STYLE [DIALOG | WINDOW | PAGE | GROUP | MENU | TABLE];
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}
```

The attributes of MENU are specified in Table 78.

Table 78 – MENU attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	MENU		✓	✓	✓	✓	✓	✓	✓
11.1.2	ACCESS		o	x	o	x	x	o	o
11.1.3	EXIT_ACTIONS		o	o	o	x	o	o	o
14.8	HELP		o	o	o	o	o	o	o
11.1.4	INIT_ACTIONS		o	o	o	x	o	o	o
11.1.5	ITEMS		m	m	m	m	m	m	m
14.9	LABEL		m	m	m	m	m	m	m
11.1.6	POST_EDIT_ACTIONS		o	x	o	x	x	o	o
11.1.7	POST_READ_ACTIONS		o	o	o	L ^a	o	o	o
11.1.8	POST_WRITE_ACTIONS		o	o	o	L ^a	o	o	o
11.1.9	PRE_EDIT_ACTIONS		o	x	o	x	x	o	o
11.1.10	PRE_READ_ACTIONS		o	o	o	L ^a	o	o	o
11.1.11	PRE_WRITE_ACTIONS		o	o	o	L ^a	o	o	o
11.1.12	STYLE		o	o	o	o	o	o	o
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	o	o	o	o
PRE_READ_ACTIONS and POST_READ_ACTIONS shall only be used on upload from device menus.									
PRE_WRITE_ACTIONS and POST_WRITE_ACTIONS shall only be used on download to device menus.									
^a Communication profiles: HART									

11.1.2 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 79.

Table 79 – ACCESS attribute

Attribute value	Description
OFFLINE	The values of the items displayed in the MENU are read locally.
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).

11.1.3 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.

Lexical structure

```
EXIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of EXIT_ACTIONS are specified in Table 82.

11.1.4 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.

Lexical structure

```
INIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of INIT_ACTIONS are specified in Table 82.

11.1.5 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.

Lexical structure

ITEMS

```

{
    menu-item-specifier [, menu-item-specifier]*
}
menu-item-specifier:=
[menu-reference [(REVIEW)]?]<cond> |
[variable-reference [(DISPLAY_VALUE | HIDDEN | READ_ONLY | NO_LABEL |
NO_UNIT)]?]<cond> |
[image-reference [(INLINE | NO_LABEL | ALIGN_LEFT | ALIGN_RIGHT)]?]<cond>
|
reference<cond> |
string-constant<cond> |
SEPARATOR<cond> |
ROWBREAK |
COLUMNBREAK

```

The MENU ITEMS are specified in Table 80.

The MENU ITEMS qualifiers are specified in Table 81.

Table 80 – MENU ITEMS

Attribute	Description
menu-reference	Reference to a MENU.
variable-reference	Reference to a VARIABLE.
image-reference	Reference to an IMAGE.
reference	Reference to a bit of a BIT_ENUMERATED VARIABLE, a BLOCK_A PARAMETERS, an element of BLOCK_A PARAMETERS, a CHART, a COLLECTION, a member of a COLLECTION, an EDIT_DISPLAY, a GRAPH, a GRID, a LIST, a member of a LIST, a METHOD, a METHOD with arguments, a PLUGIN, a RECORD, a REFERENCE_ARRAY, a member of a REFERENCE_ARRAY, a VALUE_ARRAY, or a member of a VALUE_ARRAY. Communication profiles: HART References to only an element of LIST, REFERENCE_ARRAY and VALUE_ARRAY are allowed. BLOCK_A PARAMETERS, an element of BLOCK_A PARAMETERS, a LIST, a METHOD with arguments, a RECORD, a REFERENCE_ARRAY, and a VALUE_ARRAY are not allowed.
SEPARATOR	The menu items before and after the SEPARATOR should be separated. NOTE EDD applications might not be able to distinguish between SEPARATOR and COLUMNBREAK. Either one or the other is typically used depending on the STYLE of the parent MENU.
COLUMNBREAK	Specifies a new column should be started.
ROWBREAK	Specifies a new row should be started.
string-constant	Static text that should be displayed on the menu.

For more detailed information about menu containers and contained items, see IEC 61804-4.

Table 81 – MENU ITEMS qualifiers

Attribute	Description
DISPLAY_VALUE	The value of the variable should be displayed along with its label (Limited Display EDD applications). DISPLAY_VALUE qualifier shall be assumed and applied to all VARIABLES displayed by EDD applications (Non-Limited Display EDD applications).
HIDDEN	Communication profiles: CS, GPE, PB, PN Shall only be used with data items. The data item shall be read but the item is not displayed. No space shall be reserved.
READ_ONLY	Shall only be used with VARIABLE. Specifies that the value shall be displayed but it shall not be editable from the MENU (regardless of its HANDLING attribute).
NO_LABEL	Specifies that the label of the item is not displayed; DISPLAY_VALUE is implied. Communication profiles: CS, GPE, FF, PB, PN, ISA100 Shall only be used with VARIABLE.
NO_UNIT	Shall only be used with VARIABLE. Specifies that the engineering unit of the VARIABLE shall not be displayed; DISPLAY_VALUE is implied.
REVIEW	Shall only be used with MENU. REVIEW is used to present a large quantity of data, e.g. for review purposes. Only variable-references or menu-references shall be included in the ITEMS list for menus qualified using REVIEW. All variables appearing on the menu or its sub-menus shall assume the qualifiers READ_ONLY and DISPLAY_VALUE.
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. The label of the item shall not be displayed.
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 in the row or in the column based on INLINE qualifier. If no alignment is specified, the IMAGE should be displayed centered in the row or in the column based on INLINE qualifier.
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 in the row or in the column based on INLINE qualifier. If no alignment is specified, the IMAGE should be displayed centered in the row or in the column based on INLINE qualifier.

11.1.6 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.

Lexical structure

```

POST_EDIT_ACTIONS { actions-specifier }
actions-specifier := [reference<cond> [, reference<cond>]*]?
                     [DEFINITION { compound-statement }]<cond>?

```

The attributes of POST_EDIT_ACTIONS are specified in Table 82.

Table 82 – EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS attributes

Value type	Description
reference	Communication profiles: FF, HART, ISA100 Reference to a METHOD instance. Communication profiles: CS, GPE, PB, PN Reference to a METHOD instance with arguments.
DEFINITION	Communication profiles: CS, GPE, PB, PN Lexical element (see 14.4).

11.1.7 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.

Lexical structure

```
POST_READ_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
[DEFINITION { compound-statement }]<cond>?
```

The attributes of POST_READ_ACTIONS are specified in Table 82.

11.1.8 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.

Lexical structure

```
POST_WRITE_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
[DEFINITION { compound-statement }]<cond>?
```

The attributes of POST_WRITE_ACTIONS are specified in Table 82.

11.1.9 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.

Lexical structure

```
PRE_EDIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of PRE_EDIT_ACTIONS are specified in Table 82.

11.1.10 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.

Lexical structure

```
PRE_READ_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of PRE_READ_ACTIONS are specified in Table 82.

11.1.11 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.

Lexical structure

```
PRE_WRITE_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of PRE_WRITE_ACTIONS are specified in Table 82.

11.1.12 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 Table 81) 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 83. Examples of MENU STYLES are shown in Figure 41.

Table 83 – STYLE attribute

Attribute value	Description
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. specifies that the items of the MENU should be displayed as a modal dialog box. All sub-MENUs shall retain this modal characteristic.
WINDOW	Other MENUs can be activated while this MENU is active. Interactions with all activated MENUs having WINDOW STYLE attributes are possible, i.e. specifies that the items of the MENU should be displayed as a modeless window.
PAGE	MENU items should be displayed as a tabbed property page. The MENU LABEL is displayed on the tab.
GROUP	MENU items should be displayed as a group box. The MENU's LABEL is affixed to the top of the group box. A GROUP can include COLUMNBREAK thus spanning multiple columns.
MENU	MENU items should be displayed as either a pulldown or popup menu.
TABLE	MENU items should be displayed in vertical columns. The sub-menus should be cascaded and the user allowed to navigate the resulting menu structure. Limited Display EDD applications may coerce all menus to STYLE TABLE.

For more detailed information about menu containers and contained items, see IEC 61804-4.

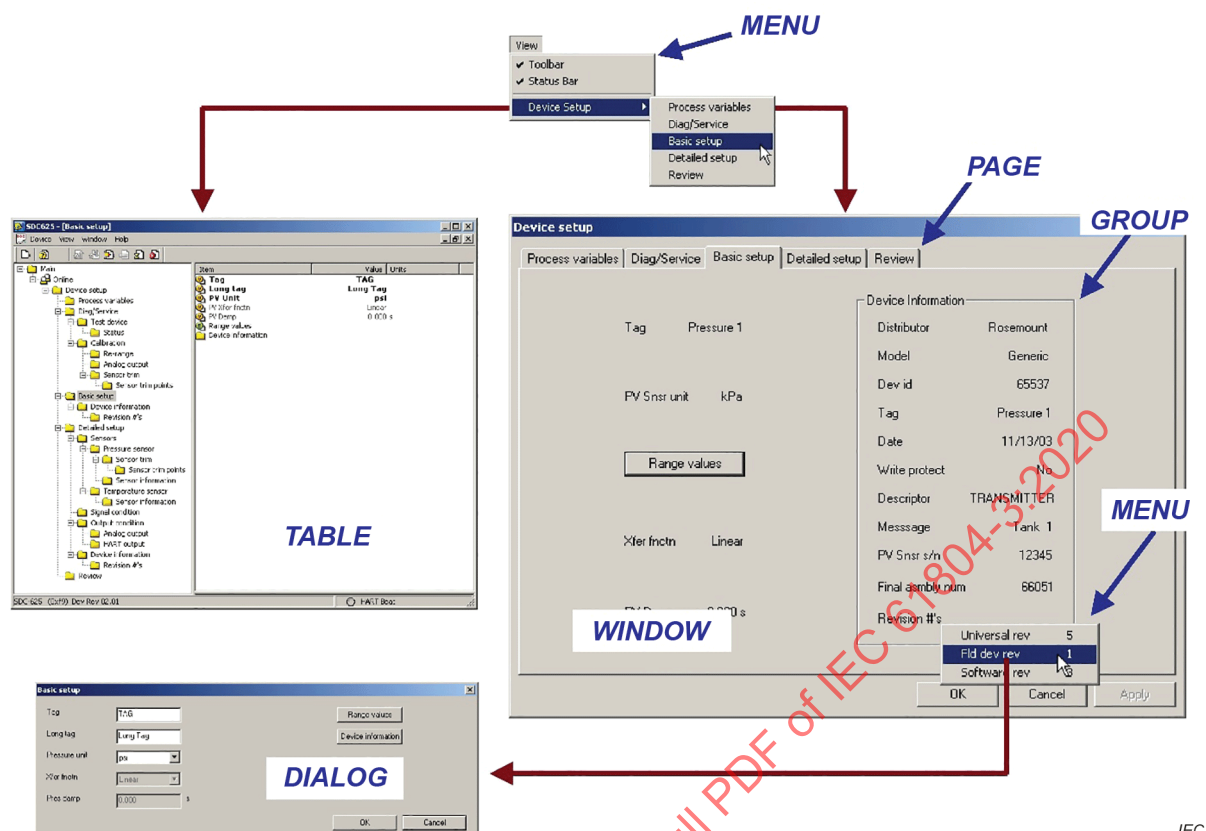


Figure 41 – Examples of MENU STYLES

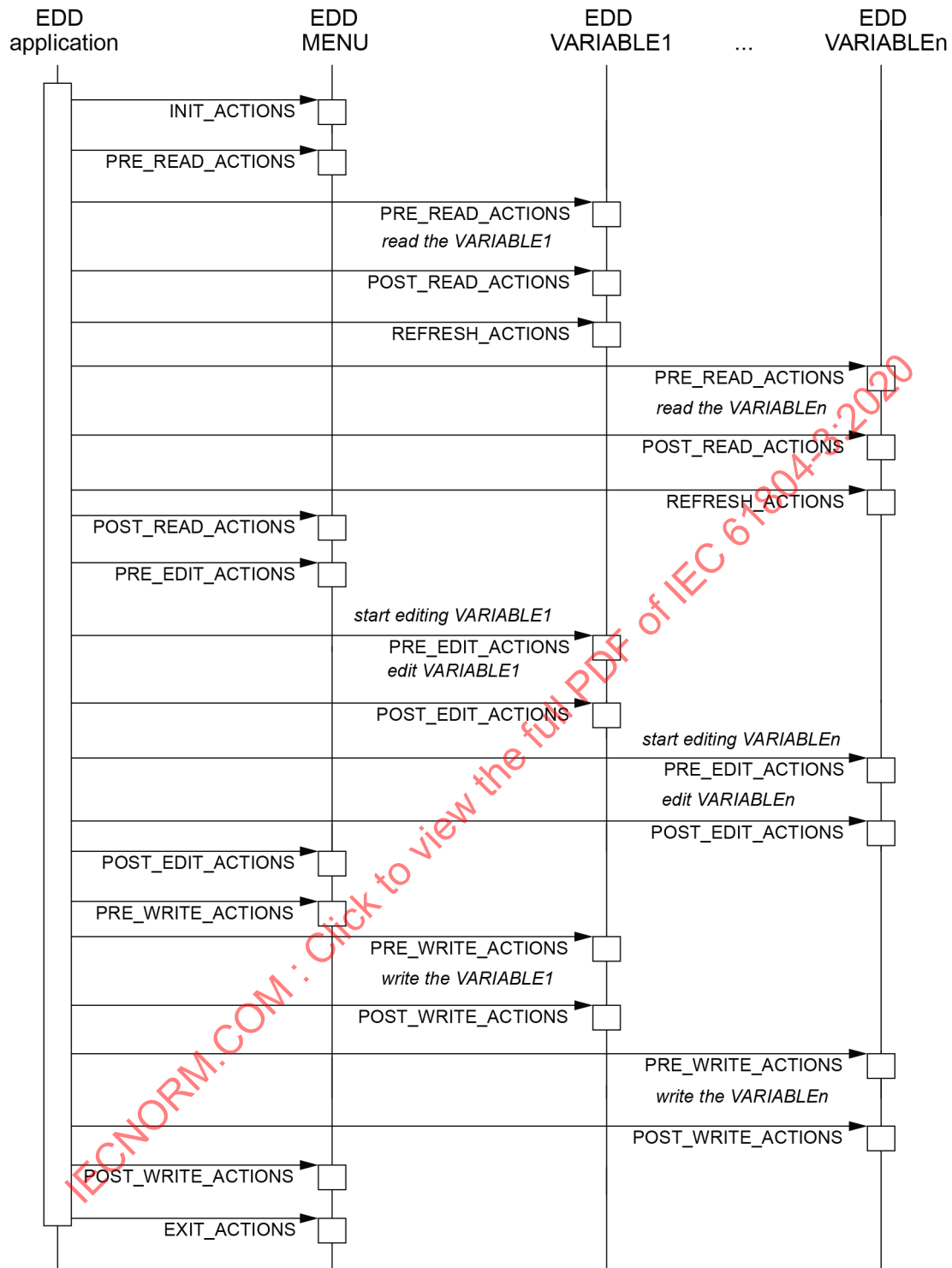
11.1.13 Sequence diagrams for actions

Figure 42 shows 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.



IEC

Figure 42 – MENU activation

11.2 EDIT_DISPLAY

11.2.1 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. EDIT_DISPLAYs are presented in the same way as a modal dialog (i.e. a MENU with STYLE DIALOG).

Lexical structure

EDIT_DISPLAY *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  DISPLAY_ITEMS { reference+ | reference<cond>+ }
  EDIT_ITEMS { reference+ | reference<cond>+ }
  POST_EDIT_ACTIONS { actions-specifier }
  PRE_EDIT_ACTIONS { actions-specifier }
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}
```

The attributes of EDIT_DISPLAY are specified in Table 84.

Table 84 – EDIT_DISPLAY attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		EDIT_DISPLAY	x	✓	x	✓	✓	x	x
11.2.3		DISPLAY_ITEMS	x	o	x	o	o	x	x
11.2.2		EDIT_ITEMS	x	m	x	m	m	x	x
14.8		HELP	x	x	x	o	x	x	x
14.9		LABEL	x	m	x	m	m	x	x
11.2.4		POST_EDIT_ACTIONS	x	o	x	o	o	x	x
11.2.5		PRE_EDIT_ACTIONS	x	o	x	o	o	x	x
14.17		VALIDITY	x	x	x	o	x	x	x
14.18		VISIBILITY	x	x	x	o	x	x	x

11.2.2 EDIT_ITEMS

Purpose

The EDIT_ITEMS attribute defines the set of data items to 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*+ | *reference*<cond>+ }

The attribute of EDIT_ITEMS is specified in Table 85.

Table 85 – EDIT_ITEMS attribute

Value type	Description
reference	Communication profiles: HART Reference to a VARIABLE or a WRITE_AS_ONE instance.
reference<cond>	Communication profiles: FF, ISA100 Reference to a VARIABLE or a WRITE_AS_ONE, a BLOCK_A PARAMETERS instance or an element of BLOCK_A PARAMETERS. May be conditionally specified.

11.2.3 DISPLAY_ITEMS

Purpose

The DISPLAY_ITEMS attribute defines the set of data item 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*+ | *reference*<cond>+ }

The attribute of DISPLAY_ITEM is specified in Table 86.

Table 86 – DISPLAY_ITEM attributes

Value type	Description
reference	Communication profiles: HART Reference to a VARIABLE.
reference<cond>	Communication profiles: FF, ISA100 Reference to a VARIABLE or a WRITE_AS_ONE, a BLOCK_A PARAMETERS instance or an element of BLOCK_A PARAMETERS. May be conditionally specified.

11.2.4 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.

Lexical structure

```

POST_EDIT_ACTIONS
{
    reference<cond> [, reference<cond>]*
}

```

The attributes of POST_EDIT_ACTIONS are specified in Table 87.

11.2.5 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.

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<cond> [, reference<cond>]*
}

```

The attributes of PRE_EDIT_ACTIONS are specified in Table 87.

Table 87 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS attribute

Attribute	Description
reference	Reference to a METHOD instance or a METHOD instance with arguments.

11.3 IMAGE

11.3.1 General structure

Purpose

The IMAGE basic construct is used to display images on the user interface.

Lexical structure

```

IMAGE identifier
{
    LABEL string-value;
    HELP string-value;
    LINK reference<cond>;
    PATH string-literal<cond>;
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
}

```

The attributes of IMAGE are specified in Table 88.

Table 88 – IMAGE attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	IMAGE		✓	✓	✓	✓	✓	✓	✓
14.8	HELP		o	o	o	o	o	o	o
14.9	LABEL		o	o	o	o	o	o	o
11.3.2	LINK		o	o	o	o	o	o	o
11.3.3	PATH		m	m	m	m	m	m	m
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	o	o	o	o

Usage

Figure 43 shows an image definition.

```
IMAGE Sensor_Diagram
{
  LABEL "Sensor Diagram";
  HELP "This Diagram shows the sensor characteristic";
  PATH "SenDiaEn.jpg\
        |en zz|SenDiaEn_LowRes.jpg\
        |de|SenDiaDe.jpg\
        |de zz|SenDiaDe_LowRes.jpg";
}
```

Figure 43 – EDD example of an IMAGE

An IMAGE definition can be used to invoke a METHOD or a MENU of STYLE DIALOG or WINDOW with the LINK attribute.

Figure 44 shows an EDD example of using the LINK attribute in an image.

```
MENU
{
  ITEMS
  {
    ...,
    Self_Test_Image (INLINE),
    ...
  }
}

IMAGE Self_Test_Image
{
  LABEL "Self Test";
  HELP "This Method execute the self test function of the device which needs some time";
  PATH "SelfTestImage.jpg";
  LINK Self_Test_Menu;
}

MENU Self_Test_Menu
{
  ...
}
```

Figure 44 – EDD example of an IMAGE with the LINK attribute

11.3.2 LINK

Purpose

The LINK attribute specifies an EDD item that can be accessed from the IMAGE. This allows additional information to be provided about the IMAGE. Once the IMAGE is displayed, selecting the image causes the hyperlinked EDD item to be activated.

A referenced MENU shall be of style DIALOG or WINDOW. In either case it shall be rendered as style DIALOG.

Lexical structure

LINK *reference*<cond>;

The attribute of LINK is specified in Table 89.

Table 89 – LINK attribute

Value type	Description
reference<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Reference to either a MENU, METHOD, or PLUGIN instance, may be conditionally specified
reference<cond>	Communication profiles: HART Reference to either a MENU, METHOD, PLUGIN, or EDIT_DISPLAY instance.

11.3.3 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*<cond>;

The attribute of PATH is specified in Table 90.

Table 90 – PATH attribute

Value type	Description
file-name<cond>	String literal that identifies the filename of the image, may be conditionally specified. Table 91 specifies the allowed image formats.

Image formats in Table 91 are allowed to be used within an IMAGE.

Table 91 – Image formats

Format	Description
JPG, JPEG	Format according to ISO/IEC 10918-1
PNG	Portable Network Graphics format according to ISO/IEC 15948
GIF	Graphics Interchange Format NOTE Some EDD applications may not support GIF animation.

11.4 GRID

11.4.1 General structure

Purpose

GRID describes a matrix of data 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 either horizontally (rows) or vertically (columns) as specified by the ORIENTATION attribute.

Lexical structure

GRID *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  HANDLING handling-specifier;
  HEIGHT height-width-specifier;
  ORIENTATION orientation-specifier;
  VALIDITY boolean-specifier;
  VECTORS { vectors-specifier }
  VISIBILITY boolean-specifier;
  WIDTH height-width-specifier;
}
```

The attributes of GRID are specified in Table 92.

Table 92 – GRID attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		GRID	✓	✓	✓	✓	✓	✓	✓
14.6		HANDLING	o	o	o	o	o	o	o
14.7		HEIGHT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
11.4.2		ORIENTATION	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
11.4.3		VECTORS	m	m	m	m	m	m	m
14.18		VISIBILITY	o	o	o	o	o	o	o
14.19		WIDTH	o	o	o	o	o	o	o

Usage

Figure 45 shows a GRID example.

```

VARIABLE PeakType
{
    LABEL "Peak Type";
    CLASS DEVICE;
    TYPE ENUMERATED
    {
        { 1, "False Echo"},
        { 2, "Button Echo" },
        { 3, "Unknown" }
    }
}

ARRAY arrPeakType
{
    NUMBER_OF_ELEMENTS 10;
    TYPE PeakType;
}

VARIABLE PeakDistance
{
    LABEL "Peak Distance";
    CLASS DEVICE;
    TYPE FLOAT
    {
        DEFAULT_VALUE 1.0;
    }
}

ARRAY arrPeakDistance
{
    LABEL "Peak Distance";
    NUMBER_OF_ELEMENTS 10;
    TYPE PeakDistance;
}

VARIABLE PeakAmplitude
{
    LABEL "Device Amplitude";
    CLASS DEVICE;
    TYPE FLOAT
    {
        DEFAULT_VALUE 1.0;
    }
}

ARRAY arrPeakAmplitude
{
    LABEL "Peak Amplitudes";
    NUMBER_OF_ELEMENTS 10;
    TYPE PeakAmplitude;
}

MENU DeviceEcho
{
    LABEL "Device Echo";
    ITEMS
    {
        EchoCurve,
        FoundEcho,
        FalseEchos
    }
}

MENU FoundEcho
{
    LABEL "Found echoes";
    STYLE PAGE;
    ITEMS
    {
        GridFoundEcho,
        RegisterFalseEchoes
    }
}

```

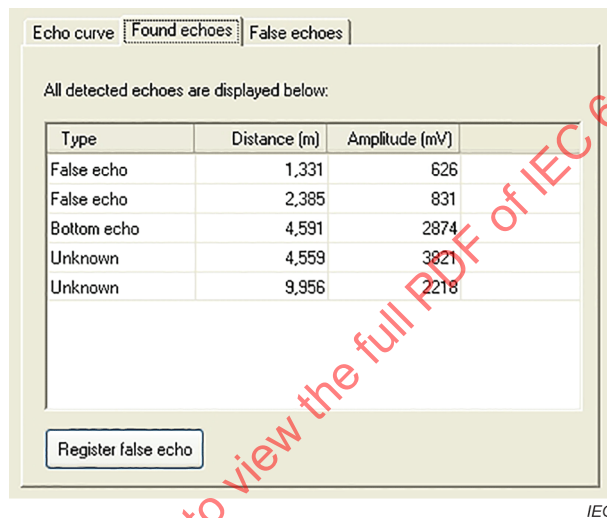
```

GRID GridFoundEchoes
{
    LABEL "All detected echoes are displayed below";
    VALIDITY IF (varModelCode == 4711) { TRUE; } ELSE { FALSE; }
    VECTORS
    {
        {"Type", arrPeakType},
        {"Distance (m)", arrPeakDistance},
        {"Amplitude (mV)", arrPeakAmplitude}
    }
}

```

Figure 45 – EDD example of a GRID

Figure 46 shows the result of the EDD example above. The constant units or units are not displayed with the values when displayed in the GRID. In Figure 46, the unit indication visible in the column heading (e.g. "(m)") is part of the VECTORS declaration.



Type	Distance (m)	Amplitude (mV)
False echo	1,331	626
False echo	2,385	831
Bottom echo	4,591	2874
Unknown	4,559	3821
Unknown	9,956	2218

Figure 46 – Result of the EDD example

11.4.2 ORIENTATION

Purpose

The ORIENTATION attribute specifies the direction in which the vectors are displayed. When no ORIENTATION attribute is specified, it defaults to VERTICAL and the VECTORS are displayed vertically.

Lexical structure

```

ORIENTATION orientation-specifier;
orientation-specifier := [ [HORIZONTAL | VERTICAL] | [HORIZONTAL |
VERTICAL] <cond> ]

```

The attributes of ORIENTATION are specified in Table 93.

Table 93 – ORIENTATION attribute

Attribute value	Description
[HORIZONTAL VERTICAL]<cond>	Communication Profiles: CS, FF, GPE, ISA100, PB, PN Vectors are displayed in rows (HORIZONTAL) or in columns (VERTICAL), may be conditionally specified
[HORIZONTAL VERTICAL]	Communication Profiles: HART Vectors are displayed in rows (HORIZONTAL) or in columns (VERTICAL)

11.4.3 VECTORS

Purpose

The VECTORS attribute specifies the contents of the GRID. The VECTORS are displayed in the order they appear. 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. The descriptions, while mandatory, may be an empty string in which case the heading is blank. If all descriptions are empty strings, then no headings cells are shown.

Lexical structure

```
VECTORS { vectors-specifier }
vectors-specifier:=
{
  [vector-specifier ,]<cond>+
}
vector-specifier:=
[description [, [string-constant | double-constant | float-constant |
integer-constant | unsigned-integer-constant]]+] |
[description [, item-reference]]+ |
[description , array-reference] |
```

The attributes of VECTORS are specified in Table 94.

Table 94 – VECTORS attribute

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

11.5 CHART

11.5.1 General structure

Purpose

CHART describes a chart used to display data from a device in the EDD application as it changes over time. A CHART is used to display continuous data values from the device, as opposed to a GRAPH that is used to display a static data set. One or more SOURCES are specified for a CHART to display by the EDD application. 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.

The background color of a CHART display is determined by the EDD application. To provide a common look and feel, it is recommended that the background color is white. It is recommended that an EDD application supports at least six curves to be displayed simultaneously.

The EDD can provide label and help information for CHARTs and its curves and axis. To build legends and help information for the curves the EDD application shall support the following rules defined in 11.5.6.

EDDL applications that do not support the graphical view of the CHART should show the related variables in a standard numerical manner. For example, the data may be shown in tabular form. The format should be chosen by the EDD application developer.

Lexical structure

CHART *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  CYCLE_TIME integer-expression | integer-expression<cond>;
  HEIGHT height-width-specifier;
  LENGTH integer-expression | integer-expression<cond>;
  MEMBERS { members-specifier }
  TYPE chart-type-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  WIDTH height-width-specifier;
}
```

The attributes of CHART are specified in Table 95.

Table 95 – CHART attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	CHART		✓	✓	✓	✓	✓	✓	✓
11.5.2		CYCLE_TIME	o	o	o	o	o	o	o
14.7		HEIGHT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
11.5.3		LENGTH	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
11.5.4		TYPE	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o
14.19		WIDTH	o	o	o	o	o	o	o

Usage

Figure 47 shows a chart with one curve in a dialog. The curve is refreshed in a cycle time of 1 s.

```

MENU measuring_values
{
    LABEL "Measuring Values";
    STYLE DIALOG;
    ITEMS
    {
        primary_value_view
    }
}

MENU primary_value_view
{
    LABEL "Primary Measuring Value";
    STYLE PAGE;
    ITEMS
    {
        primary_value_chart
    }
}

VARIABLE primary_value
{
    LABEL "Primary Value";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "Bar";
}

SOURCE primary_value_source
{
    MEMBERS
    {
        PRIM_VAL, primary_value;
    }
}

CHART primary_value_chart
{
    LABEL "Primary Value";
    MEMBERS
    {
        CHART1, primary_value_source;
    }
}

```

Figure 47 – Example of a chart with one curve in a dialog

Figure 48 is an EDD example that shows the combining of two SOURCES. Figure 49 shows how an EDD application may display the chart.


```

VARIABLE primary_value
{
    LABEL "Primary Value";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

VARIABLE primary_value_undamped
{
    LABEL "Undamped PV";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

AXIS axis_0_14
{
    LABEL "axis 0-14";
    MIN_VALUE 0;
    MAX_VALUE 14;
}

SOURCE primary_value_stack1
{
    MEMBERS
    {
        PRIM_VAL_1, primary_value;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0x0000FF; /*BLUE*/
}

SOURCE primary_value_stack2
{
    MEMBERS
    {
        PRIM_VAL_2, primary_value_undamped;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0xFF0000; /*RED*/
}

CHART primary_value_stack_chart
{
    LABEL "Primary Value";
    MEMBERS
    {
        CHART1, primary_value_stack1, "ph1";
        CHART2, primary_value_stack2, "pH2";
    }
    TYPE VERTICAL_BAR;
}

```

Figure 48 – Example of a chart with two SOURCES

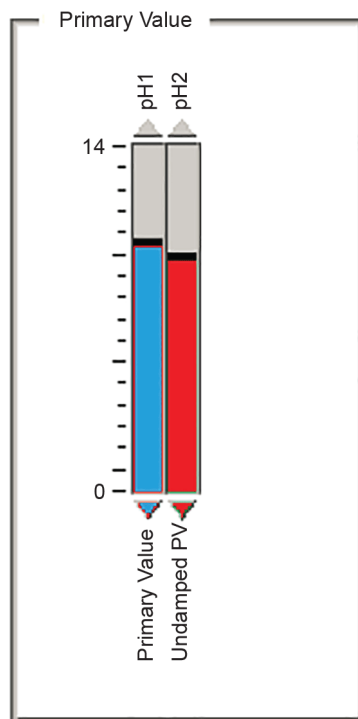


Figure 49 – Displaying example of a chart with two SOURCES

Figure 50 is an EDD example that shows the combining of three horizontal SOURCES. Figure 51 shows how an EDD application may display it.

```
VARIABLE primary_value
{
    LABEL "Primary Value";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

VARIABLE primary_value_undamped
{
    LABEL "Undamped PV";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

VARIABLE primary_value_non_temperature_compensated
{
    LABEL "PV non-TC";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

AXIS axis_0_14
{
    LABEL "axis 0-14";
    MIN_VALUE 0;
    MAX_VALUE 14;
}

AXIS axis_0_7
{
    LABEL "axis 0-7";
    MIN_VALUE 0;
    MAX_VALUE 7;
}
```

```

SOURCE primary_value_stack1
{
    MEMBERS
    {
        PRIM_VAL_1, primary_value;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0x0000FF; /*BLUE*/
}

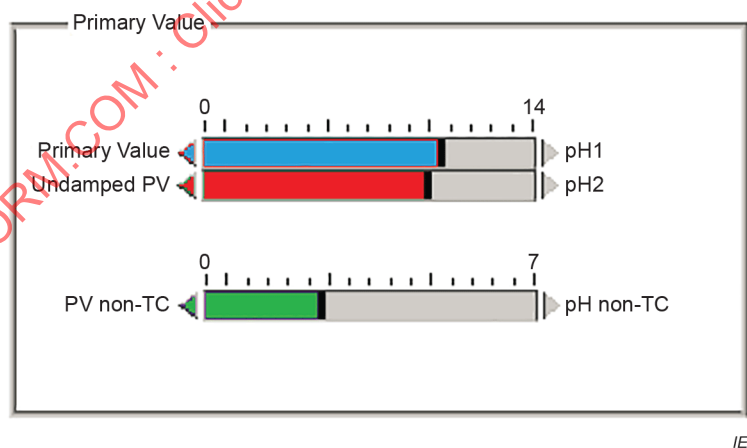
SOURCE primary_value_stack2
{
    MEMBERS
    {
        PRIM_VAL_2, primary_value_undamped;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0xFF0000; /*RED*/
}

SOURCE primary_value_single
{
    MEMBERS
    {
        PRIM_VAL_3, primary_value_non_temperature_compensated;
    }
    Y_AXIS axis_0_7;
    LINE_COLOR 0x008000; /*GREEN*/
}

CHART primary_value_stack_chart
{
    LABEL "Primary Value";
    MEMBERS
    {
        CHART1, primary_value_stack1, "pH1";
        CHART2, primary_value_single, "pH non-TC";
        CHART3, primary_value_stack2, "pH2";
    }
    TYPE HORIZONTAL_BAR;
}

```

Figure 50 – Example of a chart with three horizontal bars



IEC

Figure 51 – Displaying example of a chart with three horizontal bars

Optional INIT_ACTIONS and/or REFRESH_ACTIONS can be defined in a SOURCE.

The same method can be referenced within the INIT_ACTIONS and REFRESH_ACTIONS method lists.

Figure 52 shows an example of how to use the methods in a chart. This example shows a chart in a dialog.


```

    LINE_TYPE DATA2;
    Y_AXIS measuring_values_axis;
    MEMBERS
    {
        CALC_VAL, calculated_value;
    }
    INIT_ACTIONS    {CalculationMethod}
    REFRESH_ACTIONS {CalculationMethod}
}

CHART primary_value_chart
{
    LABEL "Primary Value";
    LENGTH 600000;
    TYPE SCOPE;
    WIDTH LARGE;
    HEIGHT SMALL;
    CYCLE_TIME 5000; /*Source value refresh of 5000 msec*/
    MEMBERS
    {
        GRAPH1, primary_value_source;
        GRAPH2, calculated_value_source;
    }
}

```

Figure 52 – Example of a chart in a dialog

11.5.2 CYCLE_TIME

Purpose

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

NOTE Due to EDD application performance reasons (e.g. the poor performance of multiplexed I/O), the update rate, latencies and jitter can cause the actual cycle times to vary. If the EDD application cannot read as fast as defined by the cycle time, it simply reads the data as fast as possible.

While CYCLE_TIME may reference a VARIABLE, the EDD application shall keep the CYCLE_TIME fixed and invariant as long as the CHART is displayed. CYCLE_TIME shall be resolved and captured immediately after the INIT_ACTIONS have been executed and before the CHART is displayed.

CYCLE_TIME shall be fixed while the CHART is displayed. If the CHART is closed and reopened, it may get a new CYCLE_TIME.

Communication profiles: HART

The lower limit for CYCLE_TIME is 1 000 ms. CYCLE_TIMES faster than that shall cause a CYCLE_TIME of 1 000 ms.

Lexical structure

CYCLE_TIME *integer-expression* | *integer-expression*<cond>;

The attribute of CYCLE_TIME is specified in Table 96.

Table 96 – CYCLE_TIME attribute

Value type	Description
integer-expression	Communication profiles: HART Length of time, in milliseconds (ms), between successive reads of the source values in the device.
integer-expression<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Length of time, in milliseconds (ms), between successive reads of the source values in the device. May be conditionally specified.

11.5.3 LENGTH

Purpose

The LENGTH attribute specifies the length of time, in milliseconds (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. While LENGTH may reference a VARIABLE, the EDD application shall keep the LENGTH fixed and invariant as long as the CHART is displayed. If the CHART is closed and reopened, it may get a new LENGTH.

Lexical structure

LENGTH *integer-expression* | *integer-expression<cond>*;

The attribute of LENGTH is specified in Table 97.

Table 97 – LENGTH attribute

Attribute	Description
integer-expression	Communication profiles: HART Length of time, in milliseconds (ms), that is displayed by the CHART. LENGTH shall resolve to a positive value greater than or equal to 60 seconds.
integer-expression<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Length of time, in milliseconds (ms), that is displayed by the CHART. May be conditionally specified.

11.5.4 TYPE

Purpose

The TYPE attribute specifies the type of CHART the EDD shall display. There are three (3) meter types and three (3) trending types supported. The meter types are GAUGE, HORIZONTAL_BAR, and VERTICAL_BAR. These are shown in Figure 54. The trending charts are: SCOPE, STRIP, and SWEEP. These are shown in Figure 53. By default, the type of a CHART without a TYPE attribute is STRIP.

For meter type charts, each SOURCE with the same Y_AXIS shall be grouped together and displayed using a single meter. The meter shall be labeled using the LABEL from the Y_AXIS. Each Y_AXIS specified within the CHART results in another meter and all meters within the CHART are then stacked together. The color of the variable's indication will be determined using the rules in 14.10. The meters are grouped together using the label of the CHART for the user's reference.

Lexical structure

TYPE *chart-type-specifier*;

Communication profiles: CS, FF, GPE, ISA100, PB, PN

```
chart-type-specifier:= [GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |  
SWEEP | VERTICAL_BAR] | [[GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |  
SWEEP | VERTICAL_BAR]<cond>]
```

Communication profiles: HART

```
chart-type-specifier:= [GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |  
SWEEP | VERTICAL_BAR] | [[GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |  
SWEEP | VERTICAL_BAR]
```

The attributes of TYPE are specified in Table 98.

Table 98 – TYPE attribute

Attribute value	Description
GAUGE	A single source value is displayed as a gauge, which is a circular, dial-type analog meter. The indicator is an arrow or pointer that rotates as the current source value varies. The minimum and maximum values are generally near the bottom of the dial and the indicator rotates clockwise as the source value increases. A GAUGE is a graphical representation of the data similar to the fuel gauge in an automobile.
HORIZONTAL_BAR	The source values are displayed as a horizontal bar chart whose length is proportional to the current value of the source. The bar is horizontally displayed with its origin (minimum value) on the left of the chart.
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 left-most beginning of the display area. The X-axis shall be updated appropriately to the display area.
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 new data added at the right end of the CHART and with the historical data shifted to the left. The oldest source values scroll off the left end of the CHART and are removed from the display area. The X-axis shall be scrolled and updated appropriately to the display area.
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.
VERTICAL_BAR	The source values are displayed as a vertical bar chart whose length is proportional to the current value of the source. The bar is vertically displayed with its origin (minimum value) at the bottom of the chart.

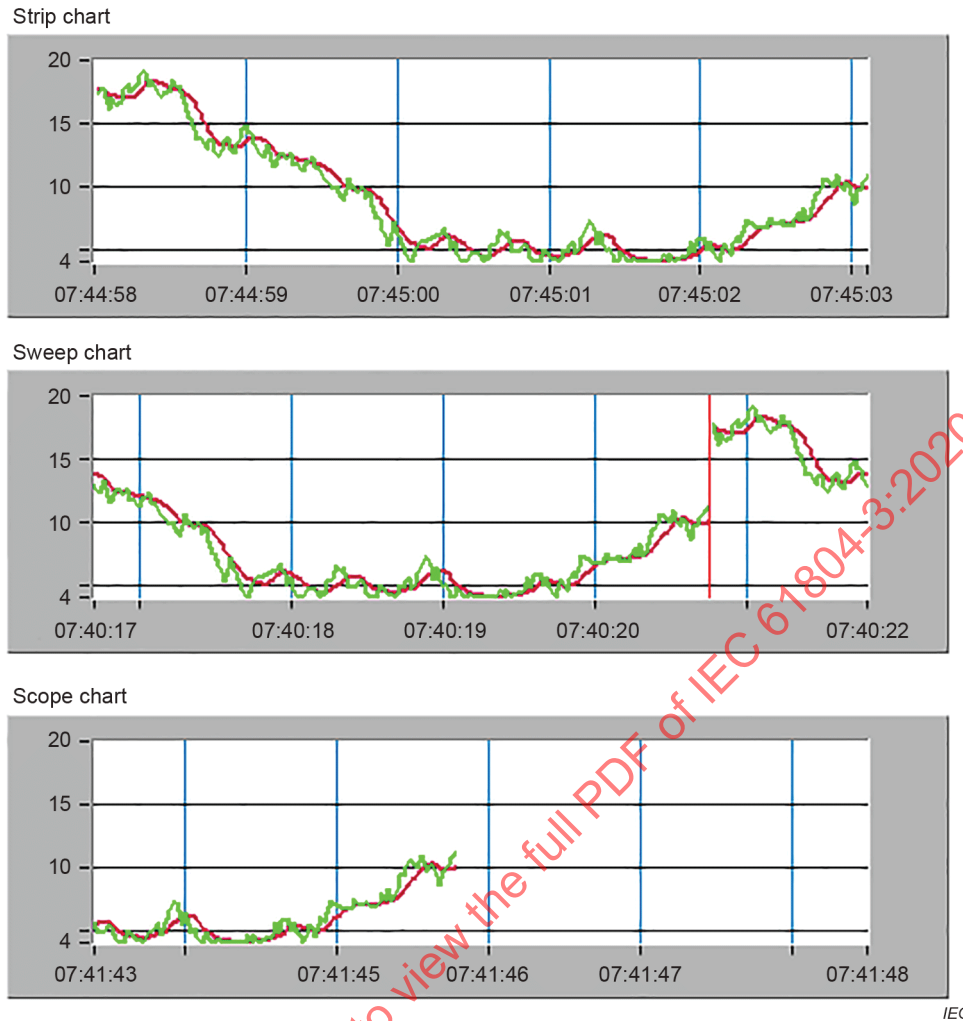


Figure 53 – Trend CHART types examples



Figure 54 – Meter CHART types examples

The LABEL of the SOURCE and VARIABLE shall be used to annotate the chart. These annotations may occur directly on the meter or appear legend-style. The label/annotation for each indicator is generated using the rules found in IEC 61804-4.

11.5.5 Sequence diagram for actions

Figure 55 shows the sequence diagram for actions related to CHARTs and their contained SOURCE members. Actions are executed and completed in sequence moving down a diagram.

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. Due to all

INIT_ACTIONS shall be executed before the CHART is displayed, CYCLE_TIME and LENGTH may be adjusted in the INIT_ACTIONS on the SOURCE members.

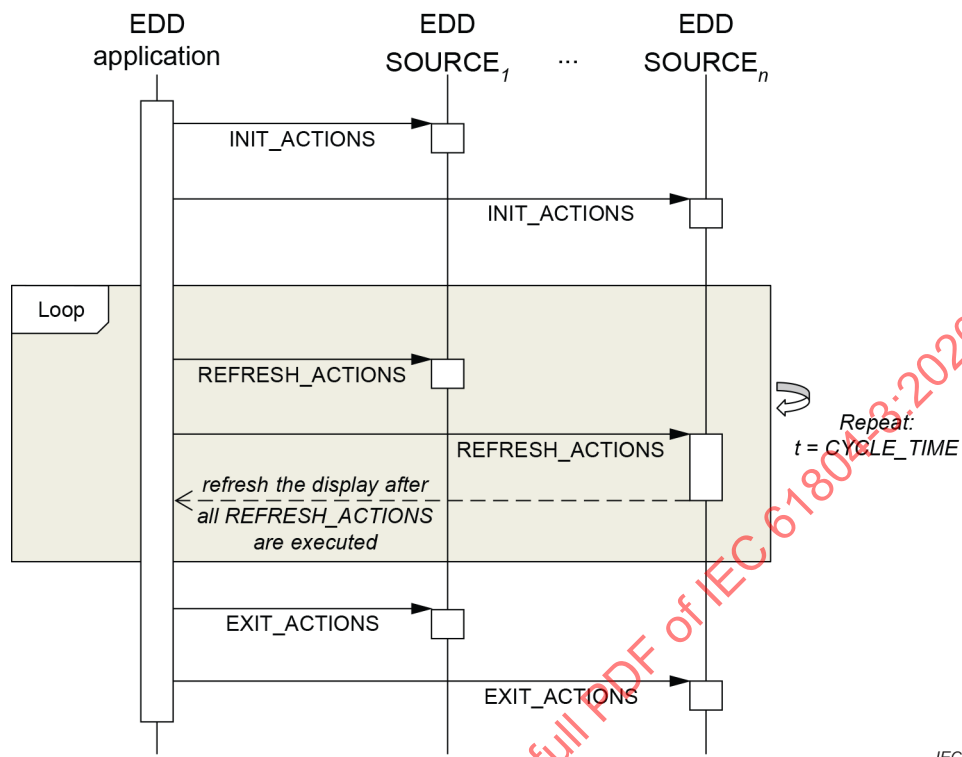


Figure 55 – CHART activation

11.5.6 CHART legend/help rules

11.5.6.1 Meter and Trend (Y_AXIS not defined) – One VARIABLE per SOURCE

- EDD application shall display the legend label/help using the CHART's member description/help.
- If the CHART's member description/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the label/help of the source reference.
- If the SOURCE's label/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the SOURCE's member description/help.
- If the SOURCE's member description/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the label/help of the VARIABLE reference (for label and help of variable reference, see also Label Concatenation for Indirect Variable References and Help Concatenation in IEC 61804-4).

11.5.6.2 Trend (Y_AXIS not defined) – Two or more VARIABLES per SOURCE

- EDD application shall display the legend label/help of a trend chart (strip, sweep and scope) the using the SOURCE's member description/help.
- If the SOURCE's member description/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the label/help of the VARIABLE reference (for label and help of variable reference, see also Label Concatenation for Indirect Variable References and Help Concatenation in IEC 61804-4).

11.5.6.3 Meter (Y_AXIS not defined) – Two or more VARIABLES per SOURCE

- EDD application shall display the legend (of the multiple sources) of a meter chart (HORIZONTAL_BAR, VERTICAL_BAR and GAUGE) using the CHART's member description/help.

- b) If the CHART's member description/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the label/help of the source reference.
- c) Furthermore, the chart shall display an additional "bar legend" label/help for each bar using the SOURCE's member description/help.
- d) If the SOURCE's member description/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the label/help of the VARIABLE reference (for label and help of variable reference, see also Label Concatenation for Indirect Variable References and Help Concatenation in IEC 61804-4).

Refer to Figure 56, which shows examples of chart legends.

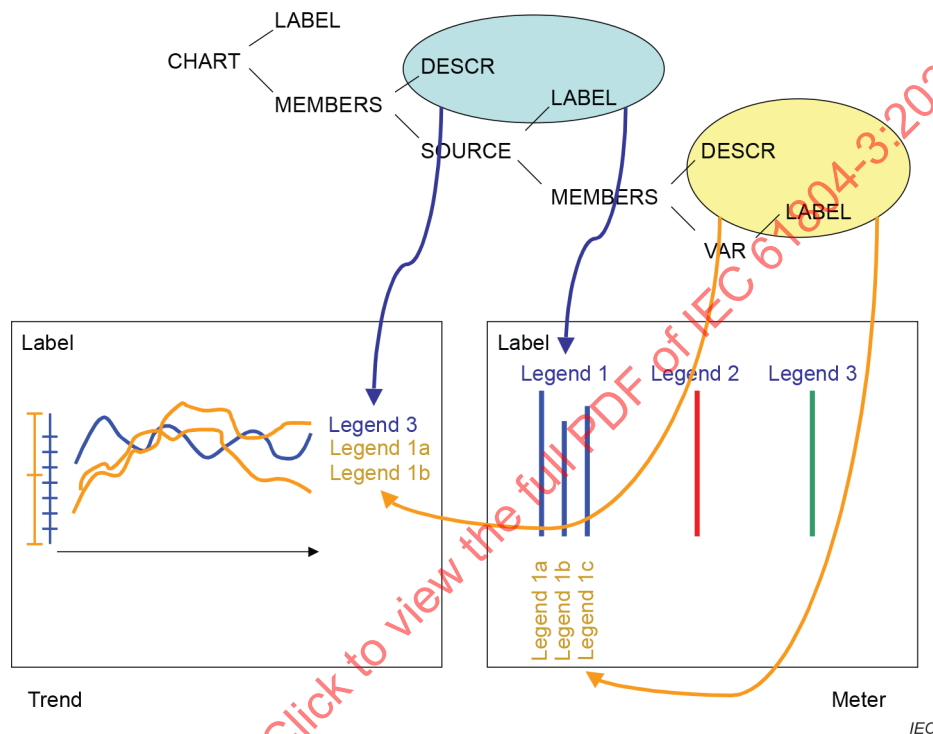


Figure 56 – Legend/help for SOURCES with single and multiple VARIABLES – Y_AXIS not defined

11.5.6.4 Trend (Y_AXIS defined) – One VARIABLE per SOURCE & two (or more) shared SOURCES

- a) Identical labeling rules as defined in 11.5.6.1.
- b) Furthermore, shall display the legend of the axis using the AXIS label.

11.5.6.5 Meter (Y_AXIS defined) – One VARIABLE per SOURCE & two (or more) shared SOURCES

- a) EDD application shall display each source with the same Y-axis, grouped together, using a single meter. For this shared axis, EDD application shall display each SOURCE either as a separate horizontal and vertical bar or as a gauge meter using a single circular or semi-circular meter
- b) EDD application shall display the legend of each meter (i.e. of the multiple sources) of a meter chart (HORIZONTAL_BAR, VERTICAL_BAR and GAUGE) using the AXIS label.
- c) Furthermore, the EDD application shall display an additional "bar/arrow" legend label/help for each bar using the CHART's member description/help.
- d) If the CHART's member description/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the label/help of the source reference.

- e) If the SOURCE's label/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the SOURCE's member description/help.
- f) If the SOURCE's member description/help is not defined or is an empty string (""), then the legend label/help shall be displayed using the label/help of the VARIABLE reference (for label and help of variable reference, see also Label Concatenation for Indirect Variable References and Help Concatenation in IEC 61804-4).

NOTE Labeling rules for 11.5.6.5 c), 11.5.6.5 d), 11.5.6.5 e) and 11.5.6.5 f) are identical to rules specified in 11.5.6.1.

Refer to Figure 57, which shows examples of chart legends.

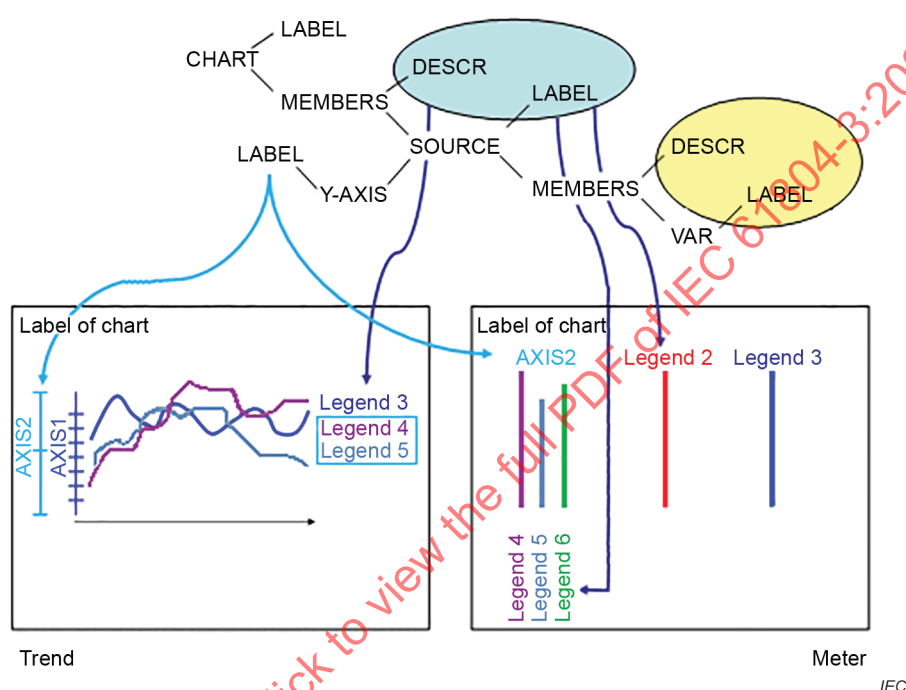


Figure 57 – Legend/help for SOURCES arrangement of multiple SOURCES with shared Y_AXIS

11.6 SOURCE

11.6.1 General structure

Purpose

A SOURCE describes a source of data values for a CHART. SOURCES have MEMBERS which identify the variable to be presented. The MEMBER shall have a name and a reference to a numeric value. Description and help are optional strings. The MEMBERS may be referenced from elsewhere in the EDD via the SOURCE identifier and the MEMBER name as specified in 16.13.

The variables that are referenced in the SOURCE should be set in the methods. The value can be calculated by using device variables and/or local variables.

If INIT_ACTIONS are defined for the SOURCE, the EDD application calls these methods instead of reading the variables.

If REFRESH_ACTIONS are defined for the SOURCE, the EDD application calls these methods cyclically in the CYCLE_TIME interval instead of reading the variables.

For trending charts (TYPES SCOPE, STRIP or SWEEP), the VARIABLE in the MEMBER list is plotted over time against the indicated Y_AXIS. For meter CHARTs (TYPES GAUGE, HORIZONTAL_BAR or VERTICAL_BAR) the variable is presented using the indicator or bar accordingly.

Lexical structure

```

SOURCE identifier
{
    LABEL string-value;
    HELP string-value;
    EMPHASIS boolean-specifier;
    EXIT_ACTIONS { actions-specifier }
    INIT_ACTIONS { actions-specifier }
    LINE_COLOR line-color-specifier;
    LINE_TYPE line-type-specifier;
    MEMBERS { members-specifier }
    REFRESH_ACTIONS { actions-specifier }
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
    Y_AXIS [reference | reference<cond>];
}

```

The attributes of SOURCE are specified in Table 99.

Table 99 – SOURCE attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		SOURCE	✓	✓	✓	✓	✓	✓	✓
14.5		EMPHASIS	o	o	o	o	o	o	o
11.6.2		EXIT_ACTIONS	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
11.6.3		INIT_ACTIONS	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.10		LINE_COLOR	o	o	o	o	o	o	o
14.11		LINE_TYPE	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
11.6.4		REFRESH_ACTIONS	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o
11.6.5		Y_AXIS	o	o	o	o	o	o	o

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

11.6.2 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.

Lexical structure

```
EXIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of EXIT_ACTIONS are specified in Table 108.

11.6.3 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, and EXIT_ACTIONS shall not be executed if a METHOD exits for an unplanned reason.

Irrespective of whether any source is displayed, the CHART shall be displayed (i.e. the CHART may be empty).

Lexical structure

```
INIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of INIT_ACTIONS are specified in Table 108.

11.6.4 REFRESH_ACTIONS

Purpose

The REFRESH_ACTIONS attribute specifies actions that are executed before the SOURCE is first drawn, when changes related to the X_AXIS or Y_AXIS are made or if a CYCLE_TIME is defined within the CHART 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 EDD application shall not draw the SOURCE value if a METHOD exits for an unplanned reason.

Irrespective of whether any source is displayed, the CHART shall be displayed (i.e. the CHART may be empty).

User interfaces embedded in REFRESH_ACTIONS METHODS should be avoided. Since, under many circumstances, refreshes can occur frequently, user interfaces embedded in REFRESH_ACTIONS can create a significant nuisance and disruption.

Lexical structure

```
REFRESH_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*] |
                    [DEFINITION { compound-statement }]<cond>
```

The attributes of REFRESH_ACTIONS are specified in Table 108.

11.6.5 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 may have its own Y_AXIS. By default, the display of the Y-axis of a CHART without a Y_AXIS attribute is inferred from the data in the SOURCE. When a Y_AXIS is not specified, the minimum and maximum values shall be large enough that all data is displayed. The minimum and maximum shall change if data is added to the CHART outside the range of the inferred AXIS.

Lexical structure

Y_AXIS [*reference* | *reference*<cond>];

The attribute of Y_AXIS is specified in Table 100.

Table 100 – Y_AXIS attribute

Attribute	Description
reference	Communication profiles: HART Reference to an AXIS instance.
reference<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Reference to an AXIS instance. May be conditionally specified.

11.7 AXIS

11.7.1 General structure

Purpose

AXIS describes an axis of a CHART or GRAPH, which specifies the frame of reference that guides the plotting of data and the interpretation of that data. Consequently MIN_VALUE, MAX_VALUE, SCALING and CONSTANT_UNIT attributes associated with the AXIS are important attributes.

Axis is optional and when not declared, the GRAPH or CHART automatically determines the axis minimum and maximum values. When an axis is not specified, the minimum and maximum values of the automatically generated axis shall be large enough that all data is displayed and the minimum and maximum shall change if data is added to the CHART or GRAPH outside the range of the automatically produced axis. The DD developer can include an AXIS reference when more control is desired.

GRAPHS and CHARTs are Cartesian and have X- and Y-axes. For a given CHART or GRAPH the X-axis is common to all data plotted and, for a CHART, the X-axis shall be time. One or more Y-axes may be associated with a CHART or GRAPH. When all data share a common Y-axis, the same AXIS construct shall be referenced from all data declarations (WAVEFORM or SOURCE). Multiple Y-axes may appear on the same CHART or GRAPH. When this is the case, the EDD application should provide indications that allow the operator to associate the AXIS with the corresponding plot.

Lexical structure

AXIS *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  CONSTANT_UNIT string-value;
  MAX_VALUE min-max-specifier;
  MIN_VALUE min-max-specifier;
  SCALING scaling-specifier;
}
```

The attributes of AXIS are specified in Table 101.

Table 101 – AXIS attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	AXIS		✓	✓	✓	✓	✓	✓	✓
11.7.4		CONSTANT_UNIT	○	○	○	○	○	○	○
14.8		HELP	○	○	○	○	○	○	○
14.9		LABEL	○	○	○	○	○	○	○
11.7.2		MAX_VALUE	○	○	○	○	○	○	○
11.7.2		MIN_VALUE	○	○	○	○	○	○	○
11.7.3		SCALING	○	○	○	○	○	○	○

11.7.2 MIN_VALUE, MAX_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.

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 first or last tick mark of the AXIS is inferred from the WAVEFORMs on the GRAPH, if MIN_VALUE or MAX_VALUE is not defined.

The min/max values for the axis may be referenced as follows:

```
axis-identifier.MIN_VALUE
axis-identifier.MAX_VALUE
axis-identifier.VIEW_MIN
axis-identifier.VIEW_MAX
```

VIEW_MIN and VIEW_MAX provide access to the current displayed maximum and minimum values displayed on the GRAPH or CHART. VIEW_MIN and VIEW_MAX are available, for example, from within a refresh method to allow the EDD to manage the device data based on the current zoom level of the GRAPH. If the same AXIS is appearing simultaneously on multiple CHARTs or GRAPHS the result of a VIEW_MIN/VIEW_MAX reference is not defined and shall not be relied upon.

Lexical structure

MAX_VALUE *min-max-specifier*;
MIN_VALUE *min-max-specifier*;
min-max-specifier := [*reference* | *double-constant* | *float-constant* | *integer-constant* | *unsigned-integer-constant*] <cond>

The attributes of MIN_VALUE and MAX_VALUE are specified in Table 102.

Table 102 – MIN_VALUE, MAX_VALUE attribute

Value type	Description
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.
double-constant	Double-precision floating-point constant.
float-constant	Single-precision floating-point constant.
integer-constant	Integer constant.
unsigned-integer-constant	Unsigned integer constant.

11.7.3 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. SCALING shall be fixed while the GRAPH or CHART is displayed. If the GRAPH or CHART is closed and reopened, the AXIS may get a new SCALING.

Lexical structure

SCALING *scaling-specifier*;
scaling-specifier := [**LINEAR** | **LOGARITHMIC**] <cond>

The attributes of SCALING are specified in Table 103.

Table 103 – SCALING attribute

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

11.7.4 CONSTANT_UNIT

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. A AXIS without a constant unit either has no units associated with it or the units are not constant. When the engineering units can vary, they shall be represented using a UNIT relation.

11.8 GRAPH

11.8.1 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 static data set, as opposed to a CHART that is used to display continuously collected data values from a device. The data series may be stored as an array or list in the device or may be data stored on behalf of the device by the EDD application using the FILE construct. In either case, one or more WAVEFORMs may be specified by a graph for display by the EDD application.

Each MEMBER shall have a name and a reference to a WAVEFORM. Description and help are optional strings and, when present are used to provide a legend for the GRAPH and context-sensitive help regarding the WAVEFORM. If not present, the LABEL and HELP for the WAVEFORM should be used for this purpose. When the EDD application begins to display a GRAPH, the WAVEFORM's specified INIT_ACTIONS and REFRESH_ACTIONS are executed. Each data series is then plotted. The EDD application allows the user to zoom in the GRAPH allowing the investigation of particular regions of interest. When this happens the new VIEW_MIN and VIEW_MAX are established for the AXIS and the REFRESH_ACTIONS are invoked allowing the EDD to manipulate or refresh data as required. The EDD application shall also allow the user to pan the graph's display region. Panning and zooming are not constrained to any limits. A legend annotating the waveforms should be available to guide the user along with labels and unit codes for the axis. To build legends and help information for the curves, the EDD application shall support following rules.

- The EDD application shall display the description if it is defined in the GRAPH members as a legend. The EDD application shall use the LABEL of the referenced WAVEFORM, if the description is not defined. An empty string constitutes a defined string.
- The EDD application shall provide help information of GRAPH members to the user. If no HELP attribute on GRAPH members exists, the help information of the WAVEFORMs shall be used.
- If the WAVEFORM's LABEL/HELP is not defined (or an empty string), then the legend label/help shall be displayed using the WAVEFORM's member description/help.
- If the WAVEFORM's member description/help is not defined (or an empty string), then the legend label/help shall be displayed with an empty string (i.e. blank).
- All legends shall be legible.

An EDD application shall combine WAVEFORMs referencing same Y_AXIS. The curves shall be shown together in the graph area and only one Y-axis shall be shown at a side of the graph area.

If the WAVEFORM data is of CLASS DYNAMIC, the GRAPH should be periodically refreshed.

The HANDLING attribute of a WAVEFORM determines whether the user can edit the waveform. When the GRAPH is closed, the EXIT_ACTIONS shall be called to allow the EDD to allow access to and manipulation of edited WAVEFORMs. The EDD application should provide a mechanism for the user to change the waveform (direct drag and click, table entry, etc.) and a mechanism for the user to indicate that the waveform modification is complete.

For a GRAPH with a CYCLE_TIME definition, the EDD application shall suspend the source value read from the device when the GRAPH is being edited.

POST_EDIT_ACTIONS on VARIABLES should be called for each change of a graph waveform. Data inputs can be checked and any data can be modified depending on a change.

The members may be referenced from elsewhere in the EDD via the GRAPH identifier and the MEMBER name using the dot notation:

graph-identifier.member-name

The background color of a GRAPH display is defined by the EDD application. To provide a common look and feel, the default background color for a GRAPH shall be WHITE. An EDD application should support the simultaneous display of at least six curves.

Figure 58 shows a graph and the visual elements, which are provided by the constructs defined in EDDL. Each of these elements will be shown in more detail in the relevant subclauses.

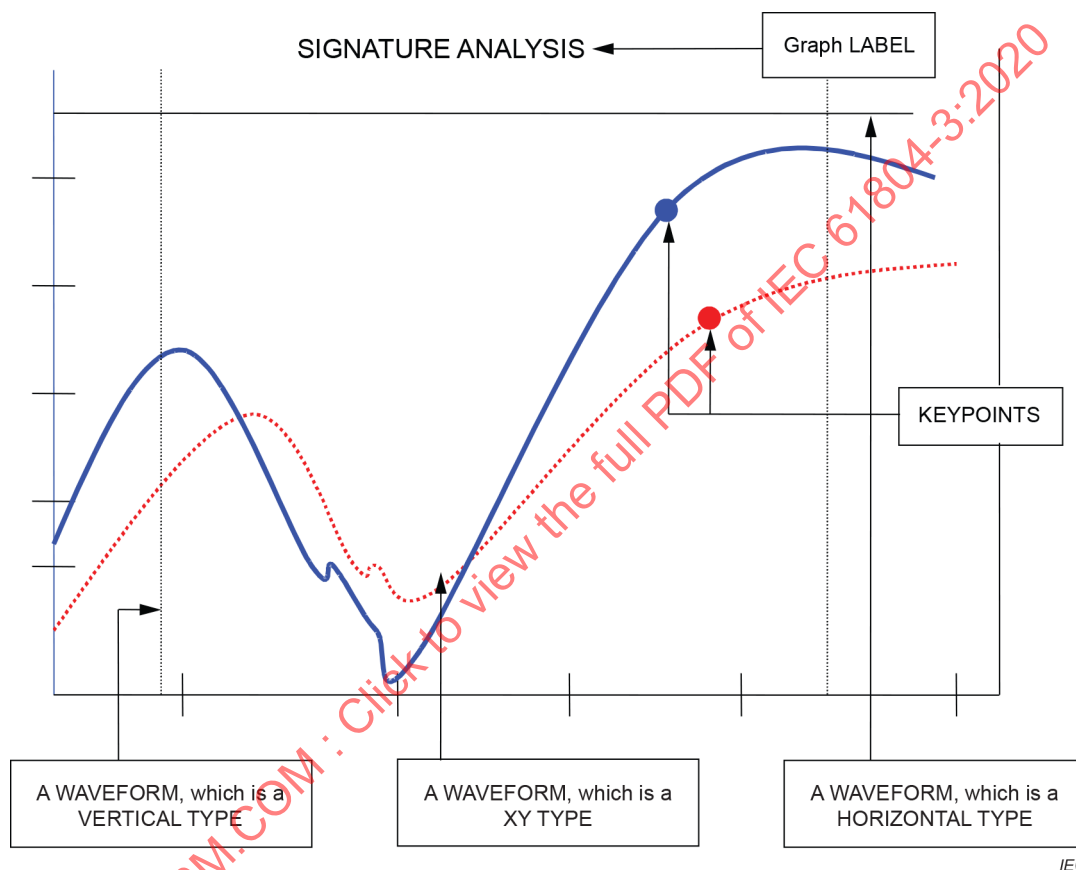


Figure 58 – A graph and the visual elements

If an EDDL application does not support the graphical view of a graph, then the values of the related variables can be shown in a table.

Lexical structure

GRAPH *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  CYCLE_TIME integer-expression | integer-expression<cond>;
  HANDLING handling-specifier;
  HEIGHT height-width-specifier;
  MEMBERS { members-specifier }
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  WIDTH height-width-specifier;
  X_AXIS reference | reference<cond>;
}
```

The attributes of GRAPH are specified in Table 104.

Table 104 – GRAPH attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		GRAPH	✓	✓	✓	✓	✓	✓	✓
11.8.2		CYCLE_TIME	o	o	o	o	o	o	o
14.7		HEIGHT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o
14.19		WIDTH	o	o	o	o	o	o	o
11.8.3		X_AXIS	o	o	o	o	o	o	o

Usage

Figure 59 shows an example of a graph.

```

VARIABLE          x_value
{
    LABEL "...";
    HELP "...";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT [degC];
}

ARRAY             x_data
{
    LABEL "x-data";
    NUMBER_OF_ELEMENTS 1000;
    TYPE x_value;
}

VARIABLE          y_value
{
    LABEL "...";
    HELP "...";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT [degC];
}

ARRAY             y_data
{
    LABEL "y-data";
    NUMBER_OF_ELEMENTS 1000;
    TYPE y_value;
}

VARIABLE          x_min_value
{
    LABEL "...";
    HELP "...";
    CLASS LOCAL;
    TYPE FLOAT;
}

AXIS x_axis_signature
{
    LABEL "x axis";
    MIN_VALUE DEFAULT_X_MIN_VALUE;
    MAX_VALUE DEFAULT_X_MAX_VALUE;
}

AXIS y_axis_signature
{
    LABEL "y axis";
    MIN_VALUE DEFAULT_Y_MIN_VALUE;
    MAX_VALUE DEFAULT_Y_MAX_VALUE;
}

WAVEFORM value_signature1
{
    TYPE          XY
    {
        X_VALUES      { x_data }
        Y_VALUES      { y_data }
    }
    INIT_ACTIONS { read_first_signature }
    REFRESH_ACTIONS { read_signature }
    Y_AXIS y_axis_signature;
}

x_max_value      LIKE x_min_value;
device_x_min_value LIKE x_min_value;
device_x_max_value LIKE x_min_value;

y_min_value      LIKE x_min_value;
y_max_value      LIKE x_min_value;
device_y_min_value LIKE x_min_value;
device_y_max_value LIKE x_min_value;

METHOD read_first_signature
{
    DEFINITION

```

```

{
    // read data from the device depending predefined default
    // MIN_VALUE and MAX_VALUE of the x and y axis
    x_min_value = DEFAULT_X_MIN_VALUE;
    x_max_value = DEFAULT_X_MAX_VALUE;
    y_min_value = DEFAULT_Y_MIN_VALUE;
    y_max_value = DEFAULT_Y_MAX_VALUE;
    WriteCommand (write_data_area);
    ReadCommand (read_data);
    x_axis_signature.MIN_VALUE = device_x_min_value;
    x_axis_signature.MAX_VALUE = device_x_max_value;
    y_axis_signature.MIN_VALUE = device_y_min_value;
    y_axis_signature.MAX_VALUE = device_y_max_value;
}

METHOD read_signature
{
    DEFINITION
    {
        // read data from the device depending of the current MIN_VALUE and MAX_VALUE
        // of the x axis
        x_min_value = x_axis_signature.VIEW_MIN;
        x_max_value = x_axis_signature.VIEW_MAX;
        y_min_value = y_axis_signature.VIEW_MIN;
        y_max_value = y_axis_signature.VIEW_MAX;
        WriteCommand (write_data_area);
        ReadCommand (read_data);
        if (device_x_min_value < x_min_value)
            x_axis_signature.VIEW_MIN = device_x_min_value;
        if (device_x_max_value > x_max_value)
            x_axis_signature.VIEW_MAX = device_x_max_value;
        if (device_y_min_value < y_min_value)
            y_axis_signature.VIEW_MIN = device_y_min_value;
        if (device_y_max_value > y_max_value)
            y_axis_signature.VIEW_MAX = device_y_max_value;
    }
}

GRAPH valve_signature
{
    MEMBERS
    {
        VAL_SIG, value_signature1;
    }
    X_AXIS x_axis_signature;
}

```

Figure 59 – Example of a graph

Figure 60 shows an EDD example of a common axis for WAVEFORM w1 and w2 and a second axis for WAVEFORM w3.

```

AXIS    x1 { }
AXIS    y1 { }
AXIS    y2 { }

GRAPH graph1
{
    MEMBERS
    {
        w1, w1;
        w2, w2;
        w3, w3;
    }
    X_AXIS x1;
}

WAVEFORM w1
{
    Y_AXIS y1;
    TYPE XY
    {
        Y_VALES {w1_values}
        X_INCREMENT 1
        X_INITIAL 0
    }
}

WAVEFORM w2
{
    Y_AXIS y1;
    TYPE XY
    {
        Y_VALES {w2_values}
        X_INCREMENT 1
        X_INITIAL 0
    }
}

WAVEFORM w3
{
    Y_AXIS y2;
    TYPE XY
    {
        Y_VALES {w3_values}
        X_INCREMENT 1
        X_INITIAL 0
    }
}

```

Figure 60 – Multiple used axes

The EDD application should support zooming and scrolling without support in the EDD. The EDD application shows fewer or more points, or data from the currently not displayed part of the WAVEFORM on the display area. In addition to that, the REFRESH_ACTIONS in the EDD can support scrolling and zooming. The EDD application calls the REFRESH_ACTIONS if the user scrolls into an area that is not stored in the waveform data. It also calls the REFRESH_ACTIONS if the user is zooming and more points should be shown. Information about the position and zooming can be read from the axis with the VIEW_MIN and VIEW_MAX attributes.

Figure 61 shows an EDD example with device supported zooming and scrolling.

```

VARIABLE      y_value
{
    LABEL "...";
    HELP "...";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT [degC];
}

ARRAY          y_data
{
    NUMBER_OF_ELEMENTS 1000;
    TYPE y_value;
}

COMMAND read_data
{
    SLOT ...; INDEX...;
    OPERATION READ;
    TRANSACTION
    {
        REPLY
        {
            device_t_min_value, device_t_max_value,
            device_y_min_value, device_y_max_value,
            y_data
        }
    }
}

COMMAND read_current_data
{
    SLOT ...; INDEX...;
    OPERATION READ;
    TRANSACTION
    {
        REPLY
        {
            device_t_min_value, device_t_max_value,
            device_y_min_value, device_y_max_value,
            y_data
        }
    }
}

COMMAND write_data_area
{
    SLOT ...; INDEX...;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            t_min_value, t_max_value,
            y_min_value, y_max_value
        }
    }
}

VARIABLE y_increment
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE TIME;
}

VARIABLE current_date_time1
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE DATE_AND_TIME;
}

VARIABLE t_min_value
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE DATE_AND_TIME;
}

```

```

}

t_max_value LIKE t_min_value;
t_device_min_value LIKE t_min_value;
t_device_max_value LIKE t_min_value;

VARIABLE y_min_value
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE FLOAT;
}

y_max_value LIKE min_value;
device_y_min_value LIKE min_value;
device_y_max_value LIKE min_value;

AXIS y_axis_signature
{
    LABEL          "...";
    HELP           "...";
    MIN_VALUE      y_min_value;
    MAX_VALUE      y_max_value;
    SCALING        LINEAR;
}

AXIS t_axis_signature
{
    LABEL          "...";
    HELP           "...";
    MIN_VALUE      t_min_value;
    MAX_VALUE      t_max_value;
    SCALING        LINEAR;
}

WAVEFORM value_signature1
{
    LABEL          "...";
    HELP           "...";
    HANDLING       READ; // alternative READ & WRITE
    LINE_TYPE      DATA1;
    EMPHASIS       TRUE;

    TYPE           XY
    {
        X_INITIAL      current_date_time1; // starting time in milliseconds
        X_INCREMENT     y_increment;        // time between two points in ms
        Y_VALUES        { y_data }
    }

    Y_AXIS          y_axis_signature;

    INIT_ACTIONS { init_signature}
    REFRESH_ACTIONS { refresh_signature}
}

METHOD init_signature
{
    DEFINITION
    {
        // read current data and store area t_device_min_value,
        // t_device_max_value, y_device_min_value, y_device_max_value
        read_command (read_current_data);

        // calculation of the waveform data
        value_signature1.X_INITIAL = t_device_max_value;
        value_signature1.X_INCREMENT = (t_device_max_value-t_device_min_value)/
            ( y_data.NUMBER_OF_ELEMENTS - 1 );

        // set the visible area to the area from the device
        t_axis_signature.MIN_VALUE = t_device_min_value;
        t_axis_signature.MAX_VALUE = t_device_max_value;
        y_axis_signature.MIN_VALUE = y_device_min_value;
        y_axis_signature.MAX_VALUE = y_device_max_value;
    }
}

METHOD refresh_signature

```



```

{
    DEFINITION
    {
        // read data from the device depending of the current
        // MIN_VALUE and MAX_VALUE of the time axis
        t_min_value = t_axis_signature.MIN_VALUE;

        t_max_value = t_axis_signature.MAX_VALUE;
        y_min_value = y_axis_signature.MIN_VALUE;
        y_max_value = y_axis_signature.MAX_VALUE;

        // writing requested t_min_value, t_max_value, y_min_value, y_max_value
        write_command (write_data_area);

        // reading points and area      t_device_min_value, t_device_max_value,
        //                               y_device_min_value, y_device_max_value from the
device
        read_command (read_data);

        // reduce the visible area to the from the device supported area
        if (t_device_min_value > t_min_value)
            t_axis_signature.MIN_VALUE = t_device_min_value;
        if (t_device_max_value < t_max_value)
            t_axis_signature.MAX_VALUE = t_device_max_value;
        if (y_device_min_value > y_min_value)
            y_axis_signature.MIN_VALUE = y_device_min_value;
        if (y_device_max_value < y_max_value)
            y_axis_signature.MAX_VALUE = y_device_max_value;
    }
}

```

Figure 61 – EDD with device-supported zooming and scrolling

11.8.2 CYCLE_TIME

Purpose

The CYCLE_TIME attribute specifies the length of time, in milliseconds (ms), the EDD application shall wait between successive reads of the source values in the device. The GRAPH shall be refreshed (and the REFRESH_ACTIONS invoked) with the periodicity specified by the CYCLE_TIME attribute.

NOTE Due to EDD application performance reasons (e.g. the poor performance of multiplexed I/O), the update rate, latencies and jitter can cause the actual cycle times to vary.

By default, a GRAPH has no cycle time. In this case, CYCLE_TIME defaults to

Communication profiles: CS, GPE, HART, PB, PN

+infinity (i.e. the GRAPH is not periodically updated)

Communication profiles: FF, ISA100

1 000 milliseconds

Otherwise, CYCLE_TIME shall resolve to a positive value. While CYCLE_TIME may reference a VARIABLE, the EDD application shall keep the CYCLE_TIME fixed and invariant as long as the GRAPH is displayed. The CYCLE_TIME shall be resolved and captured immediately after the INIT_ACTIONS have been executed and before the GRAPH is displayed.

CYCLE_TIME shall be fixed while the CHART is displayed. If the CHART is closed and reopened, it may get a new CYCLE_TIME.

Lexical structure

CYCLE_TIME *integer-expression* | *integer-expression*<cond>;

The attribute of CYCLE_TIME is specified in Table 105.

Table 105 – CYCLE_TIME attribute

Value type	Description
integer-expression	Communication profiles: HART Length of time, in milliseconds (ms), between successive reads of the source values in the device.
integer-expression<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Length of time, in milliseconds (ms), between successive reads of the source values in the device. May be conditionally specified.

11.8.3 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 may have its own Y_AXIS, or share the same 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* | *reference<cond>*;

The attribute of X_AXIS is specified in Table 106.

Table 106 – X_AXIS attribute

Value type	Description
reference	Communication profiles: HART Reference to an AXIS instance.
reference<cond>	Reference to an AXIS instance, may be conditionally specified

11.8.4 Sequence diagram for actions

Figure 62 shows the sequence diagram for actions related to GRAPHS and their contained WAVEFORM members. Actions are executed and completed in sequence moving down a diagram.

The data from the device may be adjusted prior to being displayed via the use of INIT_ACTIONS and RERESH_ACTIONS on the WAVEFORM members of the GRAPH. Because all INIT_ACTIONS shall be executed before the GRAPH is displayed, CYCLE_TIME may be adjusted in the INIT_ACTIONS on the WAVEFORM members.

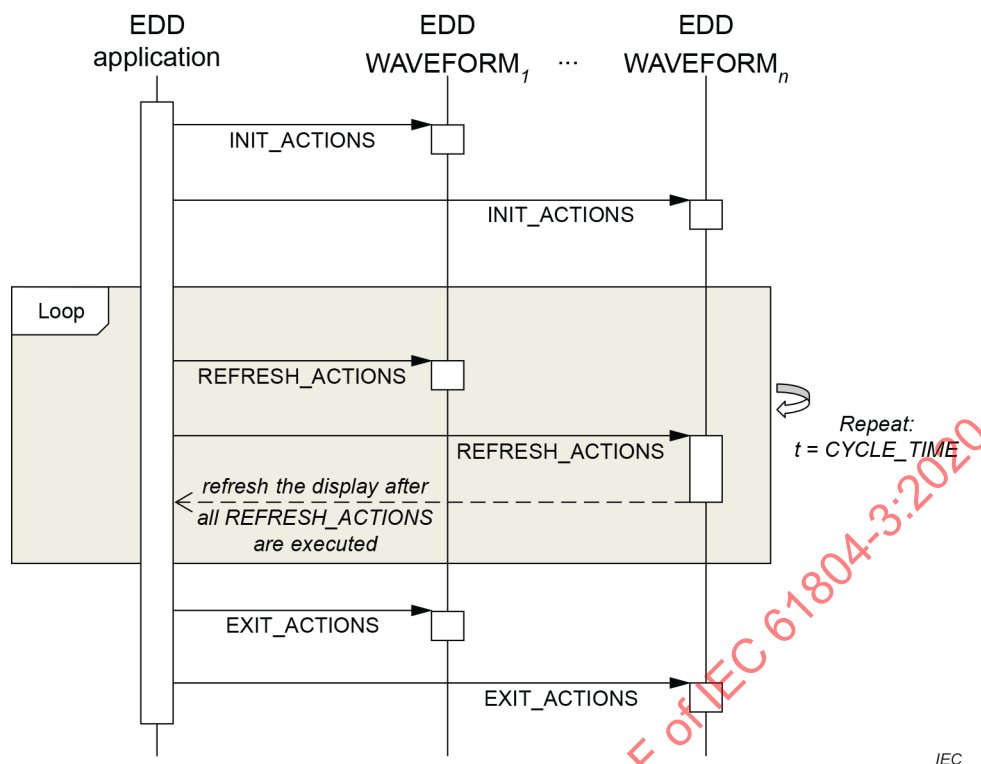


Figure 62 – GRAPH activation

11.9 WAVEFORM

11.9.1 General structure

Purpose

A WAVEFORM describes a data set that may be displayed by a GRAPH. The WAVEFORM is a vector or series of data described as X-Y data pairs, a series of Y-values equally spaced in the horizontal direction (i.e. a Y-T data series), or a horizontal or vertical line. In addition to providing the waveform data itself, key points in the data series can be highlighted.

Lexical structure

```

WAVEFORM identifier
{
  LABEL string-value;
  HELP string-value;
  EMPHASIS boolean-specifier;
  EXIT_ACTIONS { actions-specifier }
  HANDLING handling-specifier;
  INIT_ACTIONS { actions-specifier }
  KEY_POINTS
  {
    X_VALUES { key-points-specifier }
    Y_VALUES { key-points-specifier }
  }
  LINE_COLOR line-color-specifier;
  LINE_TYPE line-type-specifier;
  REFRESH_ACTIONS { actions-specifier }
  TYPE waveform-type-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  Y_AXIS [reference | reference<cond>];
}

```

The attributes of WAVEFORM are specified in Table 107.

Table 107 – WAVEFORM attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		WAVEFORM	✓	✓	✓	✓	✓	✓	✓
14.5		EMPHASIS	o	o	o	o	o	o	o
11.9.2		EXIT_ACTIONS	o	o	o	o	o	o	o
14.6		HANDLING	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
11.9.3		INIT_ACTIONS	o	o	o	o	o	o	o
11.9.4		KEY_POINTS	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.10		LINE_COLOR	o	o	o	o	o	o	o
14.11		LINE_TYPE	o	o	o	o	o	o	o
11.9.5		REFRESH_ACTIONS	o	o	o	o	o	o	o
11.9.6		TYPE	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o
11.9.7		Y_AXIS	o	o	o	o	o	o	o

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

11.9.2 EXIT_ACTIONS

Purpose

The EXIT_ACTIONS attribute specifies actions that are executed when the GRAPH containing the WAVEFORM is closed. Exit actions allow edited waveform data to be captured and manipulated as needed by the EDD. Also if the device needs to be in any special mode to supply waveform data, an EXIT_ACTION could be used to reset that mode upon closure of the graph.

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.

Lexical structure

```
EXIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of EXIT_ACTIONS are specified in Table 108.

11.9.3 INIT_ACTIONS

Purpose

The INIT_ACTIONS attribute specifies actions that are executed before the WAVEFORM is initially displayed by a GRAPH. This construct may be used for the initial reading of the WAVEFORM from the device or a FILE and also any preprocessing needed (for example, averaging or filtering of data), which is performed before the display.

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.

Irrespective of whether any waveform is displayed, the GRAPH shall be displayed (i.e. the GRAPH may be empty).

Lexical structure

```
INIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

The attributes of INIT_ACTIONS are specified in Table 108.

Table 108 – EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS attribute

Value type	Description
reference	Reference to a METHOD instance or a METHOD instance with arguments.
DEFINITION	Communication profiles: CS, GPE, PB, PN Lexical element (see 14.4).

11.9.4 KEY_POINTS

11.9.4.1 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 application. By default, a WAVEFORM without a KEY_POINTS attribute should not have any of its points highlighted.

Lexical structure

KEY_POINTS

```
{
  X_VALUES { values-specifier }
  Y_VALUES { values-specifier }
}
```

The attributes of KEY_POINTS are specified in Table 109.

Table 109 – KEY_POINTS attribute

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
11.9.4.2	X_VALUES	m	m	m	m	m	m	m
11.9.4.2	Y_VALUES	m	m	m	m	m	m	m

11.9.4.2 X_VALUES / Y_VALUES

Purpose

The X_VALUES and Y_VALUES attributes specify the X and Y coordinates 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 { values-specifier }
Y_VALUES { values-specifier }
values-specifier:= [reference | double-constant | float-constant |
                    integer-constant | unsigned-integer-constant]+
```

The attributes of X_VALUES and Y_VALUES are specified in Table 110.

Table 110 – X_VALUES, Y_VALUES attribute

Value type	Description
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.
double	Double-precision floating-point constant.
float	Single-precision floating-point constant.
integer	Integer constant.
unsigned_integer	Unsigned integer constant.

11.9.5 REFRESH_ACTIONS

Purpose

The REFRESH_ACTIONS attribute specifies actions that are executed before the WAVEFORM is first drawn, 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.

Within the method, the visible area may be calculated and set on the axis with VIEW_MIN and VIEW_MAX. All the WAVEFORMs associated with the same AXIS shall be shown with the same area in the GRAPH.

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 displayed.

Irrespective of whether any waveform is displayed the GRAPH shall be displayed (i.e. the GRAPH may be empty).

User interfaces embedded in REFRESH_ACTIONS METHODS should be avoided. Since, under many circumstances, refreshes can occur frequently, user interfaces embedded in REFRESH_ACTIONS can create a significant nuisance and disruption.

Lexical structure

```
REFRESH_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                    [DEFINITION { compound-statement }]<cond>
```

The attributes of REFRESH_ACTIONS are specified in Table 108.

11.9.6 TYPE

11.9.6.1 General structure

Purpose

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

Lexical structure

```
TYPE waveform-type-specifier;
waveform-type-specifier:= [XY | YT | HORIZONTAL | VERTICAL]
```

The attributes of TYPE are specified in Table 111.

Table 111 – TYPE attribute

Reference	Attribute	Communication profiles						
		CS	FF	GPE	HART	ISA100	PB	PN
11.9.6.4	HORIZONTAL	S	S	S	S	S	S	S
11.9.6.5	VERTICAL	S	S	S	S	S	S	S
11.9.6.2	XY	S	S	S	S	S	S	S
11.9.6.3	YT	S	S	S	S	S	S	S

11.9.6.2 XY

Purpose

The XY attribute specifies a WAVEFORM that contains a list of (x,y) points. The XY WAVEFORM type allows the plotting of an arbitrary set of data points. The data is not required to be evenly spaced.

Lexical structure

```

XY
{
  X_VALUES { values-specifier }
  Y_VALUES { values-specifier }
  NUMBER_OF_POINTS integer-expression<cond>;
}

```

The attributes of XY are specified in Table 112.

Table 112 – XY attribute

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. The attribute X_VALUES is specified in 11.9.4.2.
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. The attribute Y_VALUES is specified in 11.9.4.2.
o	NUMBER_OF_POINTS	Specifies the number of valid data points in X_VALUES and Y_VALUES. Frequently, a data series represents some sample set of data acquired by the device. Depending on the configuration and current operation of the device, the number of points acquired by the device may vary. The EDD applications shall not plot more points than indicated by the NUMBER_OF_POINTS attribute. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES and Y_VALUES.

11.9.6.3 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. The YT WAVEFORM type allows the plotting of an evenly separated or sampled set of data.

The YT WAVEFORM type shall include the X_INITIAL, X_INCREMENT, and Y_VALUES attributes. The Y coordinate of each data point is specified in the Y_VALUES list. The corresponding X coordinate is calculated using the X_INITIAL, X_INCREMENT and the offset (index) into the Y_VALUES list. In other words, the *i*-th data point's coordinates can be calculated:

```
y-coordinate = Y_VALUES[i];
x-coordinate = X_INITIAL + (X_INCREMENT * i);
```

Lexical structure

YT

```
{
  X_INITIAL [reference | double-expression | float-expression |
             integer-expression | unsigned-integer-expression]<cond>;
  X_INCREMENT [reference | double-expression | float-expression |
              integer-expression | unsigned-integer-expression]<cond>;
  Y_VALUES { values-specifier }
  NUMBER_OF_POINTS integer-expression<cond>;
}
```

The attributes of YT are specified in Table 113.

Table 113 – YT attribute

Usage	Attribute	Description
m	X_INITIAL	Specifies the X coordinate of the first point in the WAVEFORM. 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 difference between the X coordinates of adjacent points in the WAVEFORM. 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. The attribute Y_VALUES is specified in 11.9.4.2.
o	NUMBER_OF_POINTS	Specifies the number of valid data points in X_VALUES. Frequently, a data series represents some sample set of data acquired by the device. Depending on the configuration and current operation of the device, the number of points acquired by the device may vary. The EDD applications shall not plot more points than indicated by the NUMBER_OF_POINTS attribute. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES.

11.9.6.4 HORIZONTAL

Purpose

The HORIZONTAL attribute specifies a WAVEFORM that contains horizontal lines.

Lexical structure**HORIZONTAL**

```
{
  Y_VALUES { values-specifier }
}
```

The attribute of HORIZONTAL is specified in Table 114.

Table 114 – HORIZONTAL attribute

Usage	Attribute	Description
m	Y_VALUES	Specifies the Y coordinate of each horizontal line in the WAVEFORM. The attribute Y_VALUES is specified in 11.9.4.2.

11.9.6.5 VERTICAL**Purpose**

The VERTICAL attribute specifies a WAVEFORM that contains vertical lines.

Lexical structure**VERTICAL**

```
{
  X_VALUES { values-specifier }
}
```

The attribute of VERTICAL is specified in Table 115.

Table 115 – VERTICAL attribute

Usage	Attribute	Description
m	X_VALUES	Specifies the X coordinate of each vertical line in the WAVEFORM. The attribute X_VALUES is specified in 11.9.4.2.

11.9.7 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. When a Y_AXIS is not specified, the minimum and maximum values shall be large enough that all data is displayed. The minimum and maximum shall change if data is added to the GRAPH outside the range of the inferred AXIS.

Lexical structure

```
Y_AXIS [reference | reference<cond>;
```

The attribute of Y_AXIS is specified in Table 116.

Table 116 – Y_AXIS attribute

Attribute	Description
reference	Communication profiles: HART Reference to an AXIS instance.
reference<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN Reference to an AXIS instance. May be conditionally specified.

11.10 PLUGIN**11.10.1 General structure****Purpose**

PLUGIN describes a User Interface Plug-in (UIP). This allows an external entry point to be referenced from the DD using a UUID.

Lexical structure

```

PLUGIN identifier
{
  LABEL string-value;
  HELP string-value;
  UUID uuid-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}

```

The attributes of PLUGIN are specified in Table 117.

Table 117 – PLUGIN attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		PLUGIN	✓	✓	✓	✓	✓	✓	✓
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	m	m	m	m	m	m	m
11.10.2		UUID	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o

11.10.2 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. The UUID is specified by a 36 character string consisting of 32 hexadecimal digits (case-insensitive) separated into five groups separated by hyphens (8 hexadecimal digits followed by 4, 4, 4, and 12 hexadecimal digits, respectively). For example:

3F99B900-4D87-101B-99B7-AA0004007F07

NOTE This attribute can not be conditional.

Lexical structure

UUID *uuid-specifier*;

The attribute of UUID is specified in Table 118.

Table 118 – UUID attribute

Value type	Description
uuid-specifier	Unique identifier of the plug-in specified in the manifest.

12 METHODS

12.1 General structure

Purpose

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

NOTE 1 A METHOD is used to specify consistency checks, conversion algorithms between EDD variables, user acknowledges or specific device interactions such as standard operating procedures for their maintenance, calibration, commissioning, or configuration.

NOTE 2 Methods are not meant to be used to enhance device capabilities or to generate new device data. In other words, methods that generate process or status that should be provided by the device would prevent the data's use consistently outside EDD applications.

Lexical structure

```

METHOD identifier [( parameter-declaration [, parameter-declararion]* )]?
{
  LABEL string-value;
  HELP string-value;
  ACCESS [OFFLINE | ONLINE];
  CLASS method-class-specifier [& method-class-specifier]*;
  DEFINITION { compound-statement }
  PRIVATE boolean-specifier;
  TYPE method-type-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}
parameter-declaration := parameter-type [&]?identifier[[]]?

```

The attributes of METHOD are specified in Table 119.

The attribute parameter-type is specified Table 120.

Table 119 – METHOD attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		METHOD	✓	✓	✓	✓	✓	✓	✓
12.2		ACCESS	o	x	o	x	x	o	o
12.3		CLASS	o	o	o	o	o	o	o
14.4		DEFINITION	m	m	m	m	m	m	m
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.13		PRIVATE	o	o	o	o	o	o	o
12.4		TYPE	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o

Table 120 – Parameter types

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

12.2 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 121.

Table 121 – ACCESS attributes

Attribute value	Description
OFFLINE	The values of the variables used within the METHOD definition are read or written from/to the offline data set.
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.

12.3 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 method-class-specifier [& method-class-specifier]*;
:= [ALARM | ANALOG_OUTPUT | BACKGROUND | COMPUTATION | CONTAINED |
CORRECTION | DEVICE | DIAGNOSTIC | DISCRETE | FREQUENCY | HART | INPUT |
LOCAL_DISPLAY | OPERATE | OUTPUT | SERVICE | SPECIALIST | TUNE]

```

The attributes of CLASS are specified in Table 122.

Table 122 – CLASS attributes

Attribute	Description	Communication profiles
ALARM	The METHOD is associated with an alarm (e.g. specifying alarm limits, indicating alarm status, etc.)	CS, FF, GPE, ISA100, PB, PN
ANALOG_OUTPUT	The METHOD is part of an analog output block.	CS, GPE, HART, PB, PN
BACKGROUND	CLASS BACKGROUND is provided to enhance the simulation of a device using its EDD. In all EDD applications other than Simulators, this class shall be ignored. CLASS BACKGROUND METHODS shall not belong to any other CLASS. CLASS BACKGROUND shall only be used in METHODS and, when used, indicates the Simulator shall periodically execute the METHOD. All METHODS of CLASS BACKGROUND shall be run at the monotonic period specified by the VARIABLE background_period. The METHOD shall not be run during a packet handling operation (i.e. between the time the request packet arrives and the corresponding response packet has been dispatched to the communications handler). If the background_period expires during this interval then the METHOD shall be run as soon as the command response is generated.	HART
COMPUTATION	The METHOD is part of a computation block.	CS, GPE, HART, PB, PN
CONTAINED	The METHOD represents the physical characteristics of the device.	CS, GPE, PB, PN
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.	CS, GPE, HART, PB, PN
DEVICE	The METHOD specifies the physical characteristics of the device.	CS, GPE, HART, PB, PN
DIAGNOSTIC	The METHOD evaluates the device status.	CS, FF, GPE, HART, ISA100, PB, PN
DISCRETE	The METHOD supports the discrete and or digital I/O of the device.	CS, GPE, HART

Attribute	Description	Communication profiles
FREQUENCY	The METHOD supports the frequency I/O of the device.	CS, GPE, HART
HART	The METHOD is part of the HART block which characterizes the HART interface. There is only one HART block.	CS, HART
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.	CS, FF, GPE, HART, ISA100, PB, PN
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.	CS, GPE, HART, PB, PN
OPERATE	The METHOD is used to control a block's operation.	CS, FF, GPE, PB, PN
OUTPUT	The METHOD is part of the output block. The values of output parameters may be accessed by another block input.	CS, FF, GPE, ISA100, PB, PN
SERVICE	The METHOD is used when performing routine maintenance.	CS, FF, GPE, HART, ISA100, PB, PN
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.	CS, GPE, HART, ISA100, PB, PN
TUNE	The METHOD is used to tune the algorithm of a block.	CS, FF, GPE, ISA100, PB, PN
Communication profiles: HART Only SPECIALIST and BACKGROUND are recommended for use. The remainder of CLASSES that are supported by the profile are only for backward compatibility, and not recommended for use in new designs. Communication profiles: FF, ISA100 One and only one of the following classes: INPUT, OUTPUT. Additional zero or more of the remaining classes that are supported by the profile.		

12.4 TYPE

Purpose

The TYPE attribute specifies the type of the returned value.

Communication profiles: HART

The TYPE attribute of a METHOD shall only be declared for METHODS referenced exclusively from other METHODS (i.e. a reference from a MENU to a METHOD that has a declared TYPE is not allowed).

Lexical structure

```
TYPE method-type-specifier;
method-type-specifier := [ 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 123.

Table 123 – TYPE attributes

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

13 Device modelling

13.1 BLOCK_A

13.1.1 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 identifier

```
{
  LABEL string-value;
  HELP string-value;
  AXIS_ITEMS { reference [, reference]* }
  CHARACTERISTICS reference;
  CHARTS { [identifier , reference [, description [, help]? ]?;]+ }
  CHART_ITEMS { reference [, reference]* }
  COLLECTION_ITEMS { reference [, reference]* }
  EDIT_DISPLAY_ITEMS { reference [, reference]* }
  FILES { [identifier , reference [, description [, help]? ]?;]+ }
  FILE_ITEMS { reference [, reference]* }
  GRAPHS { [identifier , reference [, description [, help]? ]?;]+ }
  GRAPH_ITEMS { reference [, reference]* }
  GRIDS { [identifier , reference [, description [, help]? ]?;]+ }
  GRID_ITEMS { reference [, reference]* }
  IMAGE_ITEMS { reference [, reference]* }
  ITEM_ARRAY_ITEMS { reference [, reference]* }
  LISTS { [identifier , reference [, description [, help]? ]?;]+ }
  LIST_ITEMS { reference [, reference]* }
  LOCAL_PARAMETERS { [identifier , reference [, description [,
help]? ]?;]+ }
  MENUS { [identifier , reference [, description [, help]? ]?;]+ }
  MENU_ITEMS { reference [, reference]* }
  METHODS { [identifier , reference [, description [, help]? ]?;]+ }
  METHOD_ITEMS { reference [, reference]* }
  PARAMETERS { [identifier , reference [, description [, help]? ]?;]+ }
  PARAMETER_LISTS { [identifier , reference [, description [,
help]? ]?;]+ }
  PLUGINS { [identifier , reference [, description [, help]? ]?;]+ }
  PLUGIN_ITEMS { reference [, reference]* }
  REFRESH_ITEMS { reference [, reference]* }
  SEMANTIC_MAP_ITEMS { reference [, reference]* }
  SOURCE_ITEMS { reference [, reference]+ }
  UNIT_ITEMS { reference [, reference]* }
  WAVEFORM_ITEMS { reference [, reference]* }
  WRITE_AS_ONE_ITEMS { reference [, reference]* }
}
```

The attributes of BLOCK_A are specified in Table 124.

Table 124 – BLOCK_A attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	BLOCK_A		x	✓	x	x	✓	x	x
13.1.2		AXIS_ITEMS	x	o	x	x	o	x	x
13.1.3		CHARACTERISTICS	x	m	x	x	x	x	x
13.1.4		CHARTS	x	o	x	x	o	x	x
13.1.5		CHART_ITEMS	x	o	x	x	o	x	x
13.1.6		COLLECTION_ITEMS	x	o	x	x	o	x	x
13.1.7		EDIT_DISPLAY_ITEMS	x	o	x	x	o	x	x
13.1.8		FILES	x	o	x	x	o	x	x
13.1.9		FILE_ITEMS	x	o	x	x	o	x	x
13.1.10		GRAPHS	x	o	x	x	o	x	x
13.1.11		GRAPH_ITEMS	x	o	x	x	o	x	x
13.1.12		GRIDS	x	o	x	x	o	x	x
13.1.13		GRID_ITEMS	x	o	x	x	o	x	x
14.8		HELP	x	o	x	x	o	x	x
13.1.14		IMAGE_ITEMS	x	o	x	x	o	x	x
13.1.15		ITEM_ARRAY_ITEMS	x	o	x	x	o	x	x
14.9		LABEL	x	m	x	x	m	x	x
13.1.16		LISTS	x	o	x	x	o	x	x
13.1.17		LIST_ITEMS	x	o	x	x	o	x	x
13.1.18		LOCAL_PARAMETERS	x	o	x	x	o	x	x
13.1.19		MENUS	x	o	x	x	o	x	x
13.1.20		MENU_ITEMS	x	o	x	x	o	x	x
13.1.21		METHODS	x	o	x	x	o	x	x
13.1.22		METHOD_ITEMS	x	o	x	x	o	x	x
13.1.23		PARAMETERS	x	m	x	x	m	x	x
13.1.24		PARAMETER_LISTS	x	o	x	x	o	x	x
13.1.25		PLUGINS	x	o	x	x	o	x	x
13.1.26		PLUGIN_ITEMS	x	o	x	x	o	x	x
13.1.27		REFRESH_ITEMS	x	o	x	x	o	x	x
13.1.28		SEMANTIC_MAP_ITEMS	x	o	x	x	o	x	x
13.1.29		SOURCE_ITEMS	x	o	x	x	o	x	x
13.1.30		UNIT_ITEMS	x	o	x	x	o	x	x
13.1.31		WAVEFORM_ITEMS	x	o	x	x	o	x	x
13.1.32		WRITE_AS_ONE_ITEMS	x	o	x	x	o	x	x

13.1.2 AXIS_ITEMS

Purpose

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

Lexical structure

AXIS_ITEMS { *reference* [, *reference*]* }

The attribute of AXIS_ITEMS is specified in Table 125.

Table 125 – AXIS_ITEMS attribute

Value type	Description
reference	Reference to an AXIS instance.

13.1.3 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 126.

Table 126 – CHARACTERISTICS attribute

Value type	Description
reference	Reference to a RECORD instance.

13.1.4 CHARTS

Purpose

The CHARTS attribute specifies the CHARTs associated with the BLOCK_A, plus a description and help for each CHART. Only those CHARTs that will be referenced via a specific BLOCK_A instance (see 16.20) shall be specified.

Lexical structure

CHARTS { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

The attributes of CHARTS are specified in Table 127.

Table 127 – CHARTS attribute

Value type	Description
identifier	Identifier by which the CHART may be referenced.
reference	Reference to a CHART instance.
description	Short description of the CHART.
help	Help text for the CHART.

13.1.5 CHART_ITEMS

Purpose

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

Lexical structure

CHART_ITEMS { *reference* [, *reference*]* }

The attribute of CHART_ITEMS is specified in Table 128.

Table 128 – CHART_ITEMS attribute

Value type	Description
reference	Reference to a CHART instance.

13.1.6 COLLECTION_ITEMS

Purpose

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

Lexical structure

COLLECTION_ITEMS { *reference* [, *reference*]* }

The attribute of COLLECTION_ITEMS is specified in Table 129.

Table 129 – COLLECTION_ITEMS attribute

Value type	Description
reference	Reference to a COLLECTION instance.

13.1.7 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* [, *reference*]* }

The attribute of EDIT_DISPLAY_ITEMS is specified in Table 130.

Table 130 – EDIT_DISPLAY_ITEMS attribute

Value type	Description
reference	Reference to an EDIT_DISPLAY instance.

13.1.8 FILES

Purpose

The FILES attribute specifies the FILES associated with the BLOCK_A, plus a description and help for each FILE. Only those FILES that will be referenced via a specific BLOCK_A instance (see 16.22) shall be specified.

Lexical structure

FILES { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

The attributes of FILES are specified in Table 131.

Table 131 – FILES attribute

Value type	Description
identifier	Identifier by which the FILE may be referenced.
reference	Reference to a FILE instance.
description	Short description of the FILE.
help	Help text for the FILE.

13.1.9 FILE_ITEMS

Purpose

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

Lexical structure

FILE_ITEMS { *reference* [, *reference*]* }

The attribute of FILE_ITEMS is specified in Table 132.

Table 132 – FILE_ITEMS attribute

Value type	Description
reference	Reference to a FILE instance.

13.1.10 GRAPHS

Purpose

The GRAPHS attribute specifies the GRAPHS associated with the BLOCK_A, plus a description and help for each GRAPH. Only those GRAPHS that will be referenced via a specific BLOCK_A instance (see 16.23) shall be specified.

Lexical structure

GRAPHS { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

The attributes of GRAPHS are specified in Table 133.

Table 133 – GRAPHS attribute

Value type	Description
identifier	Identifier by which the GRAPH may be referenced.
reference	Reference to a GRAPH instance.
description	Short description of the GRAPH.
help	Help text for the GRAPH.

13.1.11 GRAPH_ITEMS

Purpose

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

Lexical structure

GRAPH_ITEMS { *reference* [, *reference*]* }

The attribute of GRAPH_ITEMS is specified in Table 134.

Table 134 – GRAPH_ITEMS attribute

Value type	Description
reference	Reference to a GRAPH instance.

13.1.12 GRIDS

Purpose

The GRIDS attribute specifies the GRIDS associated with the BLOCK_A, plus a description and help for each GRID. Only those GRIDS that will be referenced via a specific BLOCK_A instance (see 16.24) shall be specified.

Lexical structure

GRIDS { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

The attributes of GRIDS are specified in Table 135.

Table 135 – GRIDS attribute

Value type	Description
identifier	Identifier by which the GRID may be referenced.
reference	Reference to a GRID instance.
description	Short description of the GRID.
help	Help text for the GRID.

13.1.13 GRID_ITEMS

Purpose

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

Lexical structure

GRID_ITEMS { *reference* [, *reference*]* }

The attribute of GRID_ITEMS is specified in Table 136.

Table 136 – GRID_ITEMS attribute

Value type	Description
reference	Reference to a GRID instance.

13.1.14 IMAGE_ITEMS**Purpose**

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

Lexical structure

IMAGE_ITEMS { *reference* [, *reference*]* }

The attribute of IMAGE_ITEMS is specified in Table 137.

Table 137 – IMAGE_ITEMS attribute

Value type	Description
reference	Reference to an IMAGE instance.

13.1.15 ITEM_ARRAY_ITEMS**Purpose**

The ITEM_ARRAY_ITEMS attribute specifies the reference array items associated with the BLOCK_A.

Lexical structure

ITEM_ARRAY_ITEMS { *reference* [, *reference*]* }

The attribute of ITEM_ARRAY_ITEMS is specified in Table 138.

Table 138 – ITEM_ARRAY_ITEMS attribute

Value type	Description
reference	Reference to a reference array instance.

13.1.16 LISTS**Purpose**

The LISTS attribute specifies the LISTS associated with the BLOCK_A, plus a description and help for each LIST. Only those LISTS that will be referenced via a specific BLOCK_A instance (see 13.1) shall be specified.

Lexical structure

LISTS { [*identifier* , *reference* [, *description* [, *help*?]?;]+ }

The attributes of LISTS are specified in Table 139.

Table 139 – LISTS attribute

Value type	Description
identifier	Identifier by which the LIST may be referenced.
reference	Reference to a LIST instance.
description	Short description of the LIST.
help	Help text for the LIST.

13.1.17 LIST_ITEMS

Purpose

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

Lexical structure

LIST_ITEMS { *reference* [, *reference*]* }

The attribute of LIST_ITEMS is specified in Table 140.

Table 140 – LIST_ITEMS attribute

Value type	Description
reference	Reference to a LIST instance.

13.1.18 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 145.

13.1.19 MENUS

Purpose

The MENUS attribute specifies the MENUs associated with the BLOCK_A, plus a description and help for each MENU. Only those MENUs that will be referenced via a specific BLOCK_A instance (see 16.27) shall be specified.

Lexical structure

MENUS { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

The attributes of MENUS are specified in Table 141.

Table 141 – MENUS attribute

Value type	Description
identifier	Identifier by which the MENU may be referenced.
reference	Reference to a MENU instance.
description	Short description of the MENU.
help	Help text for the MENU.

13.1.20 MENU_ITEMS**Purpose**

The MENU_ITEMS attribute specifies the MENUs associated with BLOCK_A.

Lexical structure

MENU_ITEMS { *reference* [, *reference*]* }

The attribute of MENU_ITEMS is specified in Table 142.

Table 142 – MENU_ITEMS attribute

Value type	Description
reference	Reference to a MENU.

13.1.21 METHODS**Purpose**

The METHODS attribute specifies the METHODS associated with the BLOCK_A, plus a description and help for each METHOD. Only those METHODS that will be referenced via a specific BLOCK_A instance (see 16.28) shall be specified.

Lexical structure

METHODS { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

The attributes of METHODS are specified in Table 143.

Table 143 – METHODS attribute

Value type	Description
identifier	Identifier by which the METHOD may be referenced.
reference	Reference to a METHOD instance.
description	Short description of the METHOD.
help	Help text for the METHOD.

13.1.22 METHOD_ITEMS

Purpose

The METHOD_ITEMS attribute specifies the METHODS associated with BLOCK_A. All METHODS and abort METHODS for the BLOCK_A should be listed.

Lexical structure

METHOD_ITEMS { *reference* [, *reference*]* }

The attribute of METHOD_ITEMS is specified in Table 144.

Table 144 – METHOD_ITEMS attribute

Value type	Description
reference	Reference to a METHOD instance.

13.1.23 PARAMETERS

Purpose

The PARAMETERS attribute specifies the inputs, outputs, and contained values associated with the BLOCK_A, plus a description and help for each value. 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 145.

Table 145 – PARAMETERS attributes

Value type	Description
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.
reference	Reference to a VARIABLE, VALUE_ARRAY or RECORD instance.
description	Short description for the reference.
help	Help text for the reference.

13.1.24 PARAMETER_LISTS

Purpose

The PARAMETER_LISTS attribute specifies the VARIABLE_LISTs associated with a BLOCK_A that can be communicated together, 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 146.

Table 146 – PARAMETER_LISTS attributes

Value type	Description
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.
reference	Reference to a VARIABLE_LIST instance.
description	Short description of the item.
help	Help text for the item.

13.1.25 PLUGINS

Purpose

The PLUGINS attribute specifies the PLUGINS associated with the BLOCK_A, plus a description and help for each PLUGIN. Only those PLUGINS that will be referenced via a specific BLOCK_A instance (see 16.30) shall be specified.

Lexical structure

PLUGINS { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

The attributes of PLUGINS are specified in Table 147.

Table 147 – PLUGINS attribute

Value type	Description
identifier	Identifier by which the PLUGIN may be referenced.
reference	Reference to a PLUGIN instance.
description	Short description of the PLUGIN.
help	Help text for the PLUGIN.

13.1.26 PLUGIN_ITEMS

Purpose

The PLUGIN_ITEMS attribute specifies the PLUGINS associated with BLOCK_A.

Lexical structure

PLUGIN_ITEMS { *reference* [, *reference*]* }

The attribute of PLUGIN_ITEMS is specified in Table 148.

Table 148 – PLUGIN_ITEMS attribute

Value type	Description
reference	Reference to a PLUGIN instance.

13.1.27 REFRESH_ITEMS

Purpose

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

Lexical structure

REFRESH_ITEMS { *reference* [, *reference*]* }

The attribute of REFRESH_ITEMS is specified in Table 149.

Table 149 – REFRESH_ITEMS attribute

Value type	Description
reference	Reference to a REFRESH instance.

13.1.28 SEMANTIC_MAP_ITEMS

Purpose

The SEMANTIC_MAP_ITEMS attribute specifies the SEMANTIC_MAP relations associated with the BLOCK_A.

Lexical structure

SEMANTIC_MAP_ITEMS { *reference* [, *reference*]* }

The attribute of SEMANTIC_MAP_ITEMS is specified in Table 150.

Table 150 – SEMANTIC_MAP_ITEMS attribute

Value type	Description
reference	Reference to a SEMANTIC_MAP instance.

13.1.29 SOURCE_ITEMS

Purpose

The SOURCE_ITEMS attribute specifies the SOURCEs associated with the BLOCK_A.

Lexical structure

SOURCE_ITEMS { *reference* [, *reference*]+ }

The attribute of SOURCE_ITEMS is specified in Table 151.

Table 151 – SOURCE_ITEMS attribute

Value type	Description
reference	Reference to a SOURCE instance.

13.1.30 UNIT_ITEMS

Purpose

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

Lexical structure

UNIT_ITEMS { *reference* [, *reference*]* }

The attribute of UNIT_ITEMS is specified in Table 152.

Table 152 – UNIT_ITEMS attribute

Value type	Description
reference	Reference to an UNIT instance.

13.1.31 WAVEFORM_ITEMS

Purpose

The WAVEFORM_ITEMS attribute specifies the WAVEFORMs associated with the BLOCK_A.

Lexical structure

WAVEFORM_ITEMS { *reference* [, *reference*]* }

The attribute of WAVEFORM_ITEMS is specified in Table 153.

Table 153 – WAVEFORM_ITEMS attribute

Value type	Description
reference	Reference to a WAVEFORM instance.

13.1.32 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* [, *reference*]* }

The attribute of WRITE_AS_ONE_ITEMS is specified in Table 154.

Table 154 – WRITE_AS_ONE_ITEMS attribute

Value type	Description
reference	Reference to a WRITE_AS_ONE instance.

13.2 BLOCK_B

13.2.1 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 *identifier*

```
{
  NUMBER integer-constant | integer-expression;
  TYPE [PHYSICAL | TRANSDUCER | FUNCTION];
}
```

The attributes of BLOCK_B are specified in Table 155.

Table 155 – BLOCK_B attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	BLOCK_B		✓	x	✓	x	x	✓	✓
13.2.2	NUMBER		m	x	m	x	x	m	m
13.2.3	TYPE		m	x	m	x	x	m	m

13.2.2 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-constant | integer-expression;*

The attributes of NUMBER are specified in Table 156.

Table 156 – NUMBER attributes

Value type	Description
integer-constant	Number of the BLOCK_B instance of the same TYPE.
integer-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 Clause 18.

13.2.3 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 157.

Table 157 – TYPE attributes

Attribute value	Description
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.
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.
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.

13.3 LIKE

Purpose

LIKE is used to create a new EDD instance based on an existing item. For example, in some devices there may be several variables that are almost identical. LIKE makes a copy of the existing item type with all attributes.

Lexical structure

```

identifier LIKE [item-type]? reference
[ {
  attribute-redefinition-specifier*
} ]?

```

The attributes of LIKE are specified in Table 158 and Table 159.

Table 158 – LIKE overview

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	LIKE		✓	✓	✓	✓	✓	✓	✓

Table 159 – LIKE attributes

Usage	Value type	Description
m	identifier	Name of the new EDD instance.
m	reference	Reference to an EDD instance which shall be copied.
o	attribute-redefinition-specifier	Specifies whether an attribute is added, deleted or redefined (see Clause 15). 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. Communication profiles: FF, ISA100 The optional item-type shall be used.

13.4 IMPORT

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 63 shows the EDDL import mechanisms.

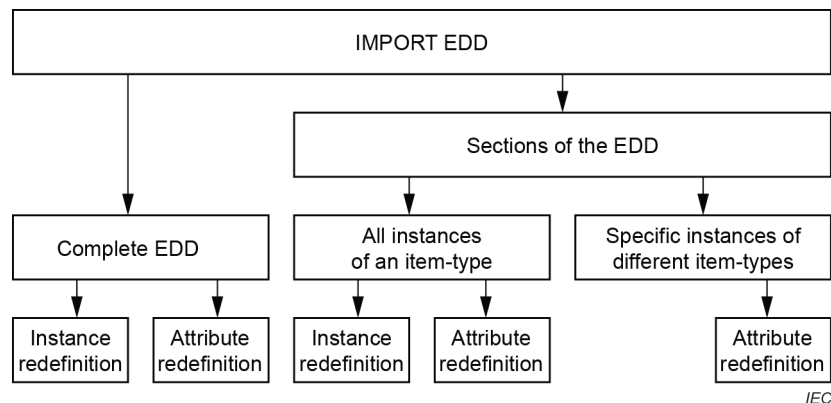


Figure 63 – EDDL import mechanisms

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

Lexical structure

Communication profiles: CS, GPE, PB, PN

```

IMPORT file-name
{
  import-keyword [& import-keyword]
  [attribute-redefinition]*
}
  
```

Communication profiles: CS, GPE, PB, PN

```

IMPORT
[DD_REVISION integer-constant ,
 DEVICE_REVISION integer-constant ,
 DEVICE_TYPE [integer-constant | identifier] ,
 [EDD_VERSION integer-constant ,]?
 MANUFACTURER [integer-constant | identifier] ,
 [MANUFACTURER_EXT string-constant]?
{
  import-keyword [& import-keyword]
  [attribute-redefinition]*
}
  
```

Communication profiles: FF, HART, ISA100

```

IMPORT
[DD_REVISION integer-constant ,
 DEVICE_REVISION integer-constant ,
 DEVICE_TYPE [integer-constant | identifier] ,
 [EDD_VERSION integer-constant ,]?
 MANUFACTURER [integer-constant | identifier] ,
{
  import-keyword [& import-keyword]
  [attribute-redefinition]*
}
  
```

```

import-keyword := [ EVERYTHING | AXES | BLOBS | BLOCKS_A | BLOCKS_B | CHARTS
| 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 | SEMANTIC_MAPS
| SOURCES | TEMPLATES | UNITS | VALUE_ARRAYS | VARIABLES | VARIABLE_LISTS |
WAVEFORMS | WRITE_AS_ONES | reference ]
  
```

The attributes are specified in Table 160 and Table 161.

Table 160 – IMPORT overview

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	IMPORT		✓	✓	✓	✓	✓	✓	✓

Table 161 – Importing device description

Attribute	Description
attribute-redefinition	See Clause 15.
AXES	All AXIS instances of an external EDD are imported.
BLOBS	All BLOB instances of an external EDD are imported.
BLOCKS_A	All BLOCK_A instances of an external EDD are imported.
BLOCKS_B	All BLOCK_B instances of an external EDD are imported.
CHARTS	All CHART instances of an external EDD are imported.
COLLECTIONS	All COLLECTION instances of an external EDD are imported.
COMMANDS	All COMMAND instances of an external EDD are imported.
COMPONENTS	All COMPONENT instances of an external EDD are imported.
COMPONENT_FOLDERS	All COMPONENT_FOLDER instances of an external EDD are imported.
COMPONENT_REFERENCES	All COMPONENT_REFERENCE instances of an external EDD are imported.
COMPONENT_RELATIONS	All COMPONENT_RELATION instances of an external EDD are imported.
DD_REVISION	See 8.2.
DELETE	An imported instance shall be deleted.
DEVICE_REVISION	See 8.3.
DEVICE_TYPE	See 8.4.
EDD_PROFILE	See 8.5.
EDD_VERSION	See 8.6.
EDIT_DISPLAYS	All EDIT_DISPLAY instances of an external EDD are imported.
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.
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.
FILES	All FILE instances of an external EDD are imported.
GRAPHS	All GRAPH instances of an external EDD are imported.
GRIDS	All GRID instances of an external EDD are imported.
IMAGES	All IMAGE instances of an external EDD are imported.
INTERFACES	All INTERFACE instances of an external EDD are imported.
LISTS	All LIST instances of an external EDD are imported.
MANUFACTURER	See 8.8.
MANUFACTURER_EXT	See 8.9.
MENUS	All MENU instances of an external EDD are imported.
METHODS	All METHOD instances of an external EDD are imported.
PLUGINS	All PLUGIN instances of an external EDD are imported.
RECORDS	All RECORD instances of an external EDD are imported.
REDEFINE	All attributes of an imported instance shall be redefined.

Attribute	Description
reference	A single instance is imported. A single instance shall not be imported, if it is already included in an item type import.
REFERENCE_ARRAYS	All REFERENCE_ARRAY instances of an external EDD are imported.
REFRESHES	All REFRESH instances of an external EDD are imported.
RELATIONS	All REFRESH, SEMANTIC_MAP, UNIT and WRITE_AS_ONE instances of an external EDD are imported.
RESPONSE_CODES	All RESPONSE_CODES instances of an external EDD are imported.
SEMANTIC_MAPS	All SEMANTIC_MAP instances of an external EDD are imported.
SOURCES	All SOURCE instances of an external EDD are imported.
TEMPLATES	All TEMPLATE instances of an external EDD are imported.
UNITS	All UNIT instances of an external EDD are imported.
VALUE_ARRAYS	All VALUE_ARRAY instances of an external EDD are imported.
VARIABLE_LISTS	All VARIABLE_LIST instances of an external EDD are imported.
VARIABLES	All VARIABLE instances of an external EDD are imported.
WAVEFORMS	All WAVEFORM instances of an external EDD are imported.
WRITE_AS_ONES	All WRITE_AS_ONE instances of an external EDD are imported.

13.5 TEMPLATE

13.5.1 General structure

Purpose

TEMPLATE describes a set of parameter values that are used for offline configuration by the EDD application to simplify the initial deployment of devices. For example, a differential pressure device may be configured as a difference pressure sensor for a heat exchanger or as a level sensor in a hydrostatic level application. There may be more than one TEMPLATE for a given device. The DEFAULT_VALUES attribute defines the default value of parameters.

During offline configuration the application of a TEMPLATE immediately changes the values in the offline dataset to be equal to the DEFAULT_VALUE defined in the TEMPLATE. This shall be followed by the evaluation of all conditionals and VALIDITY.

When a TEMPLATE is applied and the EDD application is online, the cached edit-values are updated and any values differing from the cached device-values highlighted as normal. This allows the result of applying the TEMPLATE to be reviewed prior to the user committing any changes to the device or, when offline, to the EDD application's configuration database.

Lexical structure

```

TEMPLATE identifier
{
    LABEL string-value;
    HELP string-value;
    DEFAULT_VALUES { default-values-specifier }
    VALIDITY boolean-specifier;
}

```

The attributes of TEMPLATE are specified in Table 162.

Table 162 – TEMPLATE attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		TEMPLATE	✓	✓	✓	✓	✓	✓	✓
13.5.2		DEFAULT_VALUES	m	m	m	m	m	m	m
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o

Usage

Figure 64 is an example of a TEMPLATE.

```
// Template for differential pressure
TEMPLATE diff_pressure_template
{
    HELP [diff_pressure_help];
    LABEL "Differential pressure";
    DEFAULT_VALUES
    {
        upper_range = 20.0;
        lower_range = -20.0;
        myArray[0] = 10;
    }
}
```

Figure 64 – Example of TEMPLATE usage

13.5.2 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 { default-values-specifier }
default-values-specifier:=
[reference = [expression | [expression [, expression]]+]; ]+
```

The attributes of DEFAULT_VALUES are specified in Table 163.

Table 163 – DEFAULT_VALUES attribute

Value type	Description
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.
expression	For a VARIABLE, a constant or an expression that evaluates to 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. A LIST shall be adjusted to have a COUNT that is equivalent to the number of values in the set.

EDDs with BLOCK_A shall reference data items using block referencing (see Clause 16).

13.6 COMPONENT

13.6.1 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 string-value;
    HELP string-value;
    BYTE_ORDER [BIG_ENDIAN | LITTLE_ENDIAN];
    CAN_DELETE [FALSE | TRUE]<cond>;
    CHECK_CONFIGURATION reference;
    CLASSIFICATION classification-specifier;
    COMPONENT_PARENT reference;
    COMPONENT_PATH string-constant;
    COMPONENT_RELATIONS { reference<cond> [, reference<cond>]* }
    CONNECTION_POINT reference;
    DETECT reference;
    EDD string-constant;
    INITIAL_VALUES { [initial-value-specifier]+ }
    PRODUCT_URI string-constant;
    PROTOCOL protocol-specifier;
    REDUNDANCY integer<cond>;
    SCAN reference;
    SCAN_LIST reference;
    SUPPLIED_INTERFACE { reference [, reference]* }
}

```

The attributes are specified in Table 164.

Table 164 – COMPONENT attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT	✓	x	✓	x	x	✓	✓
13.6.2		BYTE_ORDER	o	x	o	x	x	o	o
13.6.3		CAN_DELETE	o	x	o	x	x	o	o
13.6.4		CHECK_CONFIGURATION	o	x	o	x	x	o	o
14.1		CLASSIFICATION	o	x	o	x	x	o	o
14.2		COMPONENT_PARENT	o	x	o	x	x	o	o
14.3		COMPONENT_PATH	o	x	o	x	x	o	o
13.6.5		COMPONENT_RELATIONS	o	x	o	x	x	o	o
13.6.6		CONNECTION_POINT	o	x	o	x	x	o	o
13.6.7		DETECT	o	x	o	x	x	o	o
13.6.8		EDD	o	x	o	x	x	o	o
14.8		HELP	o	x	o	x	x	o	o
13.6.9		INITIAL_VALUES	o	x	o	x	x	o	o
14.9		LABEL	m	x	m	x	x	m	m
13.6.13		PRODUCT_URI	o	x	x	x	x	o	o
14.14		PROTOCOL	o	x	o	x	x	o	o
13.6.10		REDUNDANCY	o	x	o	x	x	o	o
13.6.11		SCAN	o	x	o	x	x	o	o
13.6.12		SCAN_LIST	o	x	o	x	x	o	o
14.16		SUPPLIED_INTERFACE	x	x	x	x	x	x	x

13.6.2 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 165.

Table 165 – BYTE_ORDER attribute

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

13.6.3 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] <cond>;

The attributes of CAN_DELETE are specified in Table 166.

Table 166 – CAN_DELETE attribute

Attribute value	Description
TRUE	Component can be deleted.
FALSE	Component cannot be deleted.

13.6.4 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 *reference*;

The attribute of CHECK_CONFIGURATION is specified in Table 167.

Table 167 – CHECK_CONFIGURATION attribute

Value type	Description
reference	Reference to a METHOD.

13.6.5 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 { *reference*<cond> [, *reference*<cond>]* }

The attribute of COMPONENT_RELATIONS is specified in Table 168.

Table 168 – COMPONENT_RELATIONS attribute

Value type	Description
reference	Reference to a COMPONENT_RELATION. Conditional expressions are allowed.

13.6.6 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 169.

Table 169 – CONNECTION_POINT attribute

Value type	Description
reference	Reference to a COLLECTION instance.

13.6.7 DETECT**Purpose**

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

Lexical structure

DETECT *reference*;

The attribute of DETECT is specified in Table 170.

Table 170 – DETECT attribute

Value type	Description
reference	Reference to a detect method.

13.6.8 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-literal*;

The attribute of EDD is specified in Table 171.

Table 171 – EDD attribute

Value type	Description
string-literal	File that contains the rest of the EDD.

13.6.9 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 { [initial-value-specifier]+ }
initial-value-specifier:= [reference = [expression<cond> | array-initialization]]
array-initialization:= { expression<cond> [, expression<cond>]* }

```

The attributes of INITIAL_VALUES are specified in Table 172.

Table 172 – INITIAL_VALUES attribute

Value type	Description
reference	Reference to a VARIABLE, VALUE_ARRAY or LIST.
expression	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.

13.6.10 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<cond>;

```

The attribute of REDUNDANCY is specified in Table 173.

Table 173 – REDUNDANCY attribute

Value type	Description
integer	Number of connection points.

13.6.11 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.

Lexical structure

SCAN *reference*;

The attribute of SCAN is specified in Table 174.

Table 174 – SCAN attribute

Value type	Description
reference	Reference to the scan method.

13.6.12 SCAN_LIST

Purpose

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

Lexical structure

SCAN_LIST *reference*;

The attribute of SCAN_LIST is specified in Table 175.

Table 175 – SCAN_LIST attribute

Value type	Description
reference	Reference to the scan list

13.6.13 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-constant*;

The attribute PRODUCT_URI is specified in Table 176.

Table 176 – PRODUCT_URI attribute

Value type	Description
string-constant	String literal used as reference to a Communication Server product URI.

13.7 COMPONENT_FOLDER

Purpose

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

Lexical structure

```
COMPONENT_FOLDER identifier
{
  LABEL string-value;
  HELP string-value;
  CLASSIFICATION classification-specifier;
  COMPONENT_PARENT reference;
  COMPONENT_PATH string-constant;
  PROTOCOL protocol-specifier;
}
```

The attributes of COMPONENT_FOLDER are specified in Table 177.

Table 177 – COMPONENT_FOLDER attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT_FOLDER	✓	x	✓	x	x	✓	✓
14.1		CLASSIFICATION	o	x	o	x	x	o	o
14.2		COMPONENT_PARENT	o	x	o	x	x	o	o
14.3		COMPONENT_PATH	o	x	o	x	x	o	o
14.8		HELP	o	x	o	x	x	o	o
14.9		LABEL	m	x	m	x	x	m	m
14.14		PROTOCOL	o	x	o	x	x	o	o

13.8 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.

The attribute MANUFACTURER defaults to MANUFACTURER of the EDD.

Lexical structure

```

COMPONENT_REFERENCE identifier
{
    CLASSIFICATION classification-specifier;
    COMPONENT_PARENT reference;
    COMPONENT_PATH string-constant;
    DEVICE_REVISION integer-constant;
    DEVICE_TYPE [integer-constant | identifier];
    MANUFACTURER [integer-constant | identifier];
    PROTOCOL protocol-specifier;
}

```

The attributes of COMPONENT_REFERENCE are specified in Table 178.

Table 178 – COMPONENT_REFERENCE attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT_REFERENCE	✓	x	✓	x	x	✓	✓
14.1		CLASSIFICATION	o	x	o	x	x	o	o
14.2		COMPONENT_PARENT	o	x	o	x	x	o	o
14.3		COMPONENT_PATH	o	x	o	x	x	o	o
8.3		DEVICE_REVISION	o	x	o	x	x	o	o
8.4		DEVICE_TYPE	o	x	o	x	x	o	o
8.8		MANUFACTURER	o	x	o	x	x	o	o
14.14		PROTOCOL	o	x	o	x	x	o	o

13.9 COMPONENT_RELATION

13.9.1 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. It defines the required and supplied INTERFACES for a special kind of component subcomponent relationship.

The specification of COMPONENT can be omitted if the child COMPONENT describes the relation using an appropriate PARENT_RELATION.

Lexical structure

```

COMPONENT_RELATION identifier
{
    LABEL string-value;
    HELP string-value;
    ADDRESSING { reference [, reference]* }
    COMPONENTS { component-specifier [, component-specifier]* }
    MAXIMUM_NUMBER integer<cond>;
    MINIMUM_NUMBER integer<cond>;
    RELATION_TYPE relation-specifier;
    REQUIRED_INTERFACE { reference [, reference]* }
    SUPPLIED_INTERFACE { reference [, reference]* }
}

```

The attributes of COMPONENT_RELATION are specified in Table 179.

Table 179 – COMPONENT_RELATION attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT_RELATION	✓	x	✓	x	x	✓	✓
13.9.4		ADDRESSING	o	x	o	x	x	o	o
13.9.2		COMPONENTS	m	x	m	x	x	m	m
14.8		HELP	o	x	o	x	x	o	o
14.9		LABEL	m	x	m	x	x	m	m
13.9.5		MAXIMUM_NUMBER	o	x	o	x	x	o	o
13.9.6		MINIMUM_NUMBER	o	x	o	x	x	o	o
13.9.3		RELATION_TYPE	m	x	m	x	x	m	m
13.9.7		REQUIRED_INTERFACE	x	x	x	x	x	x	x
14.16		SUPPLIED_INTERFACE	x	x	x	x	x	x	x

13.9.2 COMPONENTS

Purpose

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

Lexical structure

```

COMPONENTS { component-specifier [, component-specifier]* }
component-specifier:= [component-reference [
{ [AUTO_CREATE integer-expression<cond>]?
  [FILTER boolean-expression<cond>]?
  [REQUIRED_RANGES { required-range-specifier
                        [, required-range-specifier]* } ]?
  [MAXIMUM_NUMBER integer-expression<cond>]?
  [MINIMUM_NUMBER integer-expression<cond>]?
} ]? ]<cond>
required-range-specifier:= variable-reference
{ [MAX_VALUE [n]? expression<cond>]? [MIN_VALUE [n? expression<cond>]? }

```

The attributes of COMPONENTS are specified in Table 180.

Table 180 – COMPONENTS attribute

Attribute	Description
component-reference	COMPONENT_FOLDER or COMPONENT in the relationship.
variable-reference	Reference to a VARIABLE.
AUTO_CREATE	Number of automatically created components.
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.
MAXIMUM_NUMBER	Maximum allowed number of this component.
MINIMUM_NUMBER	Minimum required number of this component.
REQUIRED_RANGES	Variables with ranges. In a valid configuration the variables need to be in the specified range. Typically this is used to define restrictions for the addressing (e.g. slot).

Usage

The example in Figure 65 shows a module that is only allowed to plug in "slots" 0 to 4. "slot" is a variable of GasAnalyzer_module1 that is also used for addressing. The module2 can be plugged in any slot.

```
COMPONENT_RELATION GasAnalyzer_module_relation
{
  RELATION_TYPE      CHILD_COMPONENT;
  ADDRESSING         { slot }

  COMPONENTS         {
    GasAnalyzer_module1 {REQUIRED_RANGES {slot{MIN_VALUE 0; MAX_VALUE 4;}}},
    GasAnalyzer_module2
  }

  MINIMUM_NUMBER     0;
  MAXIMUM_NUMBER     10;
}
```

Figure 65 – REQUIRED_RANGES usage

13.9.3 RELATION_TYPE

Purpose

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

Lexical structure

```
RELATION_TYPE relation-specifier;
relation-specifier:= [CHILD_COMPONENT | NEXT_COMPONENT | PARENT_COMPONENT |
PREV_COMPONENT | SIBLING_COMPONENT]
```

The attributes of RELATION_TYPE are specified in Table 181.

Table 181 – RELATION_TYPE attribute

Attribute	Description
CHILD_COMPONENT	List of components that are allowed sub-modules.
PARENT_COMPONENT	List of components that are allowed parent modules or modular devices for this component.
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.
NEXT_COMPONENT	Special sibling relation to the next component related to the ADDRESSING. NEXT_COMPONENT may only be used if the ADDRESSING is specified. Not applicable for HART.
PREV_COMPONENT	Special sibling relation to the previous component related to the ADDRESSING. PREV_COMPONENT may only be used if the ADDRESSING is specified. Not applicable for HART.

Usage

The example in Figure 66 shows that the next "slot" of a module needs to be empty by specifying an empty COMPONENTS list. This COMPONENT_RELATION shall be defined in the EDD of the module where the next slot should be empty.

```

COMPONENT_RELATION GasAnalyzer_module_relation
{
    RELATION_TYPE      NEXT_COMPONENT;
    ADDRESSING          { slot } // list of addressing variables

    COMPONENTS          { }
}

```

Figure 66 – NEXT_COMPONENT usage**13.9.4 ADDRESSING****Purpose**

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

Lexical structure

ADDRESSING { *reference* [, *reference*]* }

The attribute of ADDRESSING is specified in Table 182.

Table 182 – ADDRESSING attribute

Attribute	Description
variable-reference	Reference to a variable.

13.9.5 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*<cond>;

The attribute of MAXIMUM_NUMBER is specified in Table 183.

Table 183 – MAXIMUM_NUMBER attribute

Value type	Description
integer	Total maximum allowed number of all listed components.

13.9.6 MINIMUM_NUMBER

Purpose

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

Lexical structure

MINIMUM_NUMBER *integer*<cond>;

The attribute of MINIMUM_NUMBER is specified in Table 184.

Table 184 – MINIMUM_NUMBER attribute

Value type	Description
integer	Total minimum required number of all listed components.

13.9.7 REQUIRED_INTERFACE

Purpose

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

Lexical structure

REQUIRED_INTERFACE { *reference* [, *reference*]* }

The attribute of REQUIRED_INTERFACE is specified in Table 185.

Table 185 – REQUIRED_INTERFACE attribute

Value type	Description
reference	Reference to an interface.

13.10 INTERFACE

13.10.1 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, e.g. 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 string-value;
  HELP string-value;
  DECLARATION { declaration-specifier }
}
```

The attributes of INTERFACE are specified in Table 186.

Table 186 – INTERFACE attributes

Reference	Basic construct	Attribute	Communication profiles						
			CS	FF	GPE	HART	ISA100	PB	PN
	INTERFACE		x	x	x	x	x	x	x
13.10.2		DECLARATION	x	x	x	x	x	x	x
14.8		HELP	x	x	x	x	x	x	x
14.9		LABEL	x	x	x	x	x	x	x

13.10.2 DECLARATION

Purpose

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

Lexical structure

```
DECLARATION { declaration-specifier }
declaration-specifier:= [IMAGE | MENU | METHOD | VARIABLE]*
```

The attributes of DECLARATION are specified in Table 187.

Table 187 – DECLARATION attribute

Attribute	Description
IMAGE	IMAGE declarations (see 11.3).
MENU	MENU declarations (see 11.1).
METHOD	METHOD declarations (see Clause 12).
VARIABLE	VARIABLE declarations (see 9.1).

14 Common attributes

14.1 CLASSIFICATION

Purpose

The CLASSIFICATION attribute is a single enumerated value that indicates the (possibly multiple) levels classification for this device.

Lexical structure

```

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]

```

The attributes of CLASSIFICATION are listed in Table 188.

Table 188 – CLASSIFICATION attributes

Attribute
ACTUATOR
ACTUATOR_ELECTRO_PNEUMATIC
ACTUATOR_ELECTRIC
ACTUATOR_HYDRAULIC
CONVERTER
CONTROLLER
DISCRETE_IN
DISCRETE_OUT
ELECTRICAL_DISTRIBUTION
ELECTRICAL_DISTRIBUTION_POWER_MONITORING
FREQUENCY_CONVERTER

Attribute
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-constant

14.2 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.

Lexical structure

COMPONENT_PARENT *reference*;

The attribute of COMPONENT_PARENT is specified in Table 189.

Table 189 – COMPONENT_PARENT attribute

Value type	Description
reference	Parent component in the catalog.

14.3 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.

Lexical structure

COMPONENT_PATH *string-constant*;

The attribute of COMPONENT_PATH is specified in Table 190.

Table 190 – COMPONENT_PATH attribute

Attribute	Description
string	Manufacturer specific path of the component in the catalog.

14.4 DEFINITION

Purpose

The DEFINITION attribute specifies actions to be performed and is based on a procedural programming language (for example, C programming language). Basically, a METHOD is a single function or subroutine invoked by the EDD application when selected by the user or when an event occurs that is designated to be serviced by the METHOD (e.g. a REFRESH ACTION performed when a graph is updated). 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 Clause B.1).
- Arrays.
- Operators (see Clause 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.

Communication profiles: HART

- METHODS with arguments can only be referenced within another METHOD.
- DD_STRING supports the following operators: + =

Communication profiles: FF, ISA100

- METHODS cannot be called from within METHODS.
- METHODS cannot have arguments.

Lexical structure

DEFINITION { *compound-statement* }

The attribute of DEFINITION is specified in Table 191.

Table 191 – DEFINITION attribute

Value type	Description
compound-statement	Block of instructions for a supported procedural programming language.

14.5 EMPHASIS

Purpose

The EMPHASIS attribute specifies whether or not a user interface element should be emphasized (e.g. a WAVEFORM) 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 shall not be emphasized.

Lexical structure

EMPHASIS [FALSE | TRUE] <cond>;

The attributes of EMPHASIS are specified in Table 192.

Table 192 – EMPHASIS attribute

Attribute value	Description
[FALSE TRUE]<cond>	Communication profiles: CS, FF, GPE, ISA100, PB, PN The user interface element should not be emphasized (FALSE) or should be emphasized (TRUE). May be conditionally specified.
[FALSE TRUE]	Communication profiles: HART The user interface element should not be emphasized (FALSE) or should be emphasized (TRUE).

14.6 HANDLING

Purpose

The HANDLING attribute specifies the access to the construct when referenced by the UI related construct. When the HANDLING attribute is not specified, it defaults to READ & WRITE, except for WAVEFORM which defaults to READ.

Lexical structure

HANDLING [READ | WRITE | READ & WRITE]<cond>;

The attributes of HANDLING are specified in Table 193.

Table 193 – HANDLING attribute

Attribute value	Description
READ	The construct cannot be edited.
WRITE	The construct can be edited and written to the device, but shall not be initialized from the device.
READ & WRITE	The construct can be edited and can be read from and written to the device.

14.7 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. For BIT-ENUMERATED VARIABLES, the default behavior is to display all the bits in the enumeration. If the enumeration contains more than 16 bits or if there is not enough height available to display them all, then a scrolling mechanism may be used to be able to see all of the bits. A GRAPH, a CHART, and a GRID without a HEIGHT attribute specified are handled as if it has 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 194.

Table 194 – HEIGHT/WIDTH attribute

Attribute value	Description
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.
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.
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.
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.
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.
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.
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.
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.

14.8 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-constant* | *string-constant*<cond>];

The attribute of HELP is specified in Table 195.

Table 195 – HELP attribute

Value type	Description
string-constant	Communication profiles: HART, FF, ISA100 User readable text that is not conditional. Variable references shall be limited only to those allowed for string_constant described in the formal syntax (see A.7.33).
string-constant<cond>	Communication profiles: CS, GPE, PB, PN User readable text. May be conditionally specified, and may contain all types of variable references supported by the formal syntax.

14.9 LABEL

Purpose

The LABEL attribute specifies the displayed designation of an element.

Lexical structure

LABEL [*string-constant* | *string-constant*<cond>];

The attribute of LABEL is specified in Table 196.

Table 196 – LABEL attribute

Attribute	Description
string-constant	Communication profiles: FF, ISA100, HART User readable text that is not conditional. Variable references shall be limited only to those allowed for string_constant described in the formal syntax (see A.7.33). Communication profiles: FF, ISA100 The LABEL attribute of each BLOCK_A in the EDD shall be unique. If a BLOCK_A is created via the LIKE construct, the LABEL attribute of the BLOCK_A shall be redefined.
string-constant<cond>	Communication profiles: CS, GPE, PB, PN User readable text. May be conditionally specified, and may contain all types of variable references supported by the formal syntax.

14.10 LINE_COLOR

Purpose

The LINE_COLOR attribute optionally specifies 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-expression* | *reference*<cond>]

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 197.

Table 197 – LINE_COLOR attribute

Value type	Description
integer-expression	Communication profiles: CS, FF, GPE, HART, ISA100, PB, PN Color to be used to display the user interface element: A 24-bit unsigned integer specified in hexadecimal notation (i.e. six hexadecimal characters). The most significant byte of the RGB color indicates the red intensity while the least significant byte specifies the blue intensity.
reference<cond>	Communication profiles: CS, GPE, PB, PN Reference to a VARIABLE instance containing the color to be used to display the user interface element. May be conditionally specified.

A list of standard RGB color values can be found in Standard Default Color Space for the Internet and are defined in Table 198. The predefined standard color names from the table or custom RGB values can be used in the LINE_COLOR attribute. The color names should not be used for any other meaning or purpose.

Table 198 – Standard RGB color values

	Color	Code		Color	Code
	BLACK	0x000000		GREEN	0x008000
	SILVER	0xC0C0C0		LIME	0x00FF00
	GRAY	0x808080		OLIVE	0x808000
	WHITE	0xFFFFFFFF		YELLOW	0xFFFF00
	MAROON	0x800000		NAVY	0x000080
	RED	0xFF0000		BLUE	0x0000FF
	ORANGE	0xFFA500		TEAL	0x008080
	PURPLE	0x800080		AQUA	0x00FFFF
	FUCHSIA	0xFF00FF			

14.11 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 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

Communication profiles: CS, FF, GPE, ISA100, PB, PN

```
LINE_TYPE [DATA | DATAn | LOW_LOW_LIMIT | LOW_LIMIT | HIGH_LIMIT |
HIGH_HIGH_LIMIT | TRANSPARENT] | [DATA | DATAn | LOW_LOW_LIMIT |
LOW_LIMIT | HIGH_LIMIT | HIGH_HIGH_LIMIT | TRANSPARENT]<cond>;
```

Communication profiles: HART

```
LINE_TYPE [DATA | DATAn | LOW_LOW_LIMIT | LOW_LIMIT | HIGH_LIMIT |
HIGH_HIGH_LIMIT | TRANSPARENT] | [DATA | DATAn | LOW_LOW_LIMIT |
LOW_LIMIT | HIGH_LIMIT | HIGH_HIGH_LIMIT | TRANSPARENT];
```

The attributes of LINE_TYPE are specified in Table 199.

Table 199 – LINE_TYPE attribute

Value type	Description
DATA	The user interface element contains data.
n	Integer between 1 and 9, which identifies distinct DATA line types if more than one is present.
LOW_LOW_LIMIT	The user interface element contains a low low limit.
LOW_LIMIT	The user interface element contains a low limit.
HIGH_LIMIT	The user interface element contains a high limit.
HIGH_HIGH_LIMIT	The user interface element contains a high high limit.
TRANSPARENT	For trend CHARTs and GRAPHS, only the user interface element data points are visible. For meter CHARTs, only the user interface data bar borders are visible.

14.12 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 { members-specifier }
members-specifier:= [identifier , reference [, description]? [,
help]?;]<cond>*;
```

Communication profiles: HART

Lexical structure for SOURCE MEMBERS

```
MEMBERS { members-specifier }
members-specifier:= identifier , reference [, description]? [,
help]?;
```

The attributes of MEMBERS are specified in Table 200.

Table 200 – MEMBERS attributes

Value type	Description
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.
reference	Reference to an EDD item. The type of the reference depends on the basic constructs: • CHART: reference to a SOURCE instance; • COLLECTION: Communication profiles: CS, GPE, PB, PN, FF, ISA100 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; Communication profiles: HART Reference to an EDD item; • 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 to a VARIABLE instance. • VARIABLE_LIST: reference to a RECORD, or VALUE_ARRAY or VARIABLE instance.
description	Short description of the item. Lexical structure, see 9.1.15.3.
help	Help text for the item. Lexical structure, see 9.1.15.3.

14.13 PRIVATE

Purpose

The PRIVATE attribute specifies whether an element is exposed outside the EDD application (e.g. EDD applications may support a server interface allowing access to the device from other applications). 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 201.

Table 201 – PRIVATE attributes

Attribute value	Description
FALSE	The element is public. The default value of PRIVATE is FALSE.
TRUE	The element is private.

14.14 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 *protocol-specifier*;
protocol-specifier:= [**PROFIBUS_DP** | **PROFIBUS_PA** | **PROFINET** | **FF** | **HART** | **ISA100_WCI** | *string-constant*]

The attributes of PROTOCOL are specified in Table 202.

Table 202 – PROTOCOL attribute

Attribute value	Description
FF	The component is a FOUNDATION Fieldbus component.
HART	The component is a HART component.
ISA100_WCI	The component is an ISA100 WCI component
PROFIBUS_DP	The component is a PROFIBUS DP component.
PROFIBUS_PA	The component is a PROFIBUS PA component.
PROFINET	The component is a PROFINET component.
string	Specifies any protocol. The protocol shall be supported by the used EDD application.

14.15 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 *referenced-response-codes-specifier*;
referenced-response-codes-specifier:= *reference*<cond>

The attribute of the referenced RESPONSE_CODES is specified in Table 203.

Table 203 – RESPONSE_CODES attribute

Attribute	Description
reference	Reference to a RESPONSE_CODES instance. May be conditionally specified.

Lexical structure for embedded RESPONSE_CODES

RESPONSE_CODES { *response-codes-specifier* }
response-codes-specifier:= [[*integer* , *response-code-type* , *description* , [*help*?]<cond>;]+
response-code-type:= [**DATA_ENTRY_ERROR** | **DATA_ENTRY_WARNING** | **MISC_ERROR** | **MISC_WARNING** | **MODE_ERROR** | **PROCESS_ERROR** | **SUCCESS**]

The attributes of the embedded RESPONSE_CODES are specified in Table 76.

14.16 SUPPLIED_INTERFACE

Purpose

The SUPPLIED_INTERFACE attribute lists all interfaces this component supports.

Lexical structure

SUPPLIED_INTERFACE { *reference* [, *reference*]* }

The attribute of SUPPLIED_INTERFACE is specified in Table 204.

Table 204 – SUPPLIED_INTERFACE attribute

Attribute	Description
reference	Reference to a INTERFACE instance.

14.17 VALIDITY

Purpose

Some devices can operate in several different modes. For example, a pressure transmitter might be able to calculate flow or level but, not both at the same time. Consequently, the data for level may not be available when measuring flow. The VALIDITY attribute specifies whether an element is valid or invalid. An element without a VALIDITY is always valid.

VALIDITY shall be evaluated before any use of the EDD item. For example, if VALIDITY is FALSE the EDD item does not exist and EDD item values or attributes (like VISIBILITY) are irrelevant.

VALIDITY should be expressed using a conditional IF, IF-ELSE, SELECT. If a user interface element is invalid, it should not be displayed.

When VALIDITY of a METHOD evaluates to FALSE, it has no effect (i.e. it does not prevent execution) when the METHOD is called from another method, or referenced as an ACTION.

Lexical structure

VALIDITY [**FALSE** | **TRUE**] <cond>;

The attributes of VALIDITY are specified in Table 205.

Table 205 – VALIDITY attributes

Attribute	Description
FALSE	The element is invalid.
TRUE	The element is valid.

14.18 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**] <cond>]

The attributes of VISIBILITY are specified in Table 206.

Table 206 – VISIBILITY attributes

Attribute	Description
FALSE	The element is not visible.
TRUE	The element is visible. The default value of VISIBILITY is TRUE.

14.19 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 194.

14.20 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 207.

Table 207 – WRITE_MODE attributes

Attribute value	Priority	Description
MODE_OUT_OF_SERVICE	7 – highest	Out of Service (O/S)
MODE_INITIALIZATION	6	Initialization Manual (IMan)
MODE_LOCAL_OVERRIDE	5	Local Override (LO)
MODE_MANUAL	4	Manual (Man)
MODE_AUTOMATIC	3	Automatic (Auto)
MODE_CASCADE	2	Cascade (Cas)
MODE_REMOTE_CASCADE	1	Remote-Cascade (RCas)
MODE_REMOTE_OUTPUT	0 – lowest	Remote-Output (ROut)

15 Redefinitions

15.1 Overview

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

The EVERYTHING and import-item-type import directives may import more EDD items than are actually supported by the device. As a result, in the REDEFINITIONS section these extra EDD items can be deleted using the following syntax:

```
DELETE item-type identifier;
DELETE item-type identifier;
...
...
DELETE item-type identifier;
```

where item-type is, for example, VARIABLE, ARRAY, COLLECTION, etc. In addition to deleting unwanted EDD items, the alteration list can include redefinitions of specific EDD items.

These redefinitions take the form

```
item-type identifier { redefinitions-list }
```

For example, HART Common Practice Command 48 may be truncated and is often redefined:

```
COMMAND read_additional_transmitter_status
{
    REDEFINE TRANSACTION
    {
        REQUEST {}
        REPLY
        {
            response_code,
            device_status,
            xmtr_specific_status_0,
            xmtr_specific_status_1,
            xmtr_specific_status_2,
            xmtr_specific_status_3,
            xmtr_specific_status_4,
            xmtr_specific_status_5,
            extended_device_status,
            operating_mode
        }
    }
}
```

Relations have no attributes and consequently can only be redefined as a whole. This is expressed as follows:

REDEFINE *relation-type identifier* { *relation-specifier* }

where relation-type is **UNIT**, **REFRESH**, OR **WRITE_AS_ONE**.

In general, any optional EDD item attribute may be deleted or redefined, and mandatory attributes may only be redefined. The following subclauses detail the redefinitions rules for EDDL elements. The syntax to delete or redefine an attribute is:

DELETE *attribute-type*;

REDEFINE *attribute*;

In other words, when deleting an attribute the attribute's name (LABEL, HELP, VALIDITY, etc.) is included. When redefining the keyword REDEFINE is used followed by the same attribute declaration syntax normally used to specify an EDD item. For example,

```
DELETE LABEL;
DELETE HELP;
REDEFINE VALIDITY SELECT ( measurement_type )
{
    CASE 0:  { TRUE; }
    CASE 9:  { TRUE; }
    CASE 16: { TRUE; }
    DEFAULT: { FALSE; }
}
```

Some attributes have sub-attributes or lists of attributes. Variables, reference arrays, commands and collections are examples of DD-items that have these compound declarations. For sub-attributes, the redefinition can be expressed as:

Attribute

```
{
    DELETE sub-attribute-type;
    REDEFINE sub-attribute
}
```

For example,

```
GRAPH tank_level
{
    TYPE YT
    {
        DELETE NUMBER_OF_POINTS;
        REDEFINE X_INCREMENT 0.02;
    }
}
```

or

```
VARIABLE lower_range_value
{
    TYPE FLOAT
    {
        REDEFINE DISPLAY_FORMAT ".1f";
        DELETE SCALING_FACTOR;
    }
}
```

Attributes that contain lists (like enumerators, members and response-codes) can ADD, DELETE, or REDEFINE elements of the list. These redefinitions are expressed as follows:

Attribute

```
{
    DELETE element;
    ADD element-specifier
    REDEFINE element-specifier
}
```

For example,

```
VARIABLE pressure_units
{
    TYPE ENUMERATED
    {
        ADD {7, "Bar"}
        ADD {12, "kPa"}
    }
}
```

or

```
COLLECTON pressure
{
    MEMBERS
    {
        REDEFINE DIGITAL_UNITS, pressure_range_units;
        ADD UPPER_TRIM_POINT, pressure_utp;
        ADD LOWER_TRIM_POINT, pressure_ltp;
    }
}
```

ADD DELETE and REDEFINE are the keywords used to modify the characteristics of a DD-item and thus tailor the behavior to properly modify the device. DELETE and REDEFINE can be used for most attributes while ADD allows new sub-attributes to be added to an attribute list.

The redefinition attributes are specified in Table 208.

Table 208 – Redefinition attributes

Attribute	Description
ADD	Adds the following specified attribute. "A" indicates that ADD can be applied to the attribute in the redefinition rules tables (see Table 209 to Table 238).
DELETE	Deletes the following specified attribute. "D" indicates that DELETE can be applied to the attribute in the redefinition rules tables (see Table 209 to Table 238).
REDEFINE	Redefines the following specified attribute. "R" indicates that REDEFINE can be applied to the attribute in the redefinition rules tables (see Table 209 to Table 238).

15.2 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 208 and Table 209.

Table 209 – Redefinition rules for AXIS attributes

A	D	R	Attribute	Description
	•	•	CONSTANT_UNIT	Lexical element (see 9.1.3).
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	MAX_VALUE	Lexical element (see 11.7.2).
	•	•	MIN_VALUE	Lexical element (see 11.7.2).
	•	•	SCALING	Lexical element (see 11.7.3).

15.3 BLOB

Lexical structure

BLOB *identifier*

```
{
  [[DELETE | REDEFINE] [HANDLING | HELP | IDENTITY | LABEL]] *
}
```

The redefinition rules of BLOB are specified in Table 208 and Table 210.

Table 210 – Redefinition rules for BLOB attributes

A	D	R	Attribute	Description
		•	HANDLING	Lexical element (see 14.6).
	•	•	HELP	Lexical element (see 14.8).
	•	•	IDENTITY	Lexical element (see 9.8.2).
	•	•	LABEL	Lexical element (see 14.9).

15.4 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 | ITEM_ARRAY_ITEMS | LIST_ITEMS | LOCAL_PARAMETERS |
    local-parameters-list-element | MENU_ITEMS | METHOD_ITEMS |
    PARAMETER_LISTS | parameter-lists-list-element | PLUGIN_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 208 and Table 211.

Table 211 – Redefinition rules for BLOCK_A attributes

A	D	R	Attribute	Description
		•	CHARACTERISTICS	Lexical element (see 13.1.3).
		•	LABEL	Lexical element (see 14.9).
		•	PARAMETERS	Lexical element (see 13.1.23).
•		•	parameters-list-element	Single list elements of PARAMETERS may be changed
	•	•	AXIS_ITEMS	Lexical element (see 13.1.2).
	•	•	CHART_ITEMS	Lexical element (see 13.1.5).
	•	•	COLLECTION_ITEMS	Lexical element (see 13.1.6).
	•	•	EDIT_DISPLAY_ITEMS	Lexical element (see 13.1.7).
	•	•	FILE_ITEMS	Lexical element (see 13.1.9).
	•	•	GRAPH_ITEMS	Lexical element (see 13.1.11).
	•	•	GRID_ITEMS	Lexical elements (see 13.1.13).
	•	•	HELP	Lexical element (see 14.8).
	•	•	IMAGE_ITEMS	Lexical elements (see 13.1.14).
	•	•	ITEM_ARRAY_ITEMS	Lexical element (see 13.1.15).
	•	•	LIST_ITEMS	Lexical element (see 13.1.17).
		•	LOCAL_PARAMETERS	Lexical element (see 13.1.18).
•		•	local-parameters-list-element	Single list elements of LOCAL_PARAMETERS may be changed.
	•	•	MENU_ITEMS	Lexical element (see 13.1.20).
	•	•	METHOD_ITEMS	Lexical element (see 13.1.22).
		•	PARAMETER_LISTS	Lexical element (see 13.1.23 13.1.24).
•		•	parameter-lists-list-element	Single list elements of PARAMETERS_LISTS may be changed.
	•	•	PLUGIN_ITEMS	Lexical element (see 13.1.26).
	•	•	REFRESH_ITEMS	Lexical element (see 13.1.27).
			SEMANTIC_MAP_ITEMS	Lexical element (see 13.1.28).
	•	•	SOURCE_ITEMS	Lexical element (see 13.1.29).
	•	•	UNIT_ITEMS	Lexical element (see 13.1.30).
	•	•	WAVEFORM_ITEMS	Lexical element (see 13.1.31).
	•	•	WRITE_AS_ONE_ITEMS	Lexical element (see 13.1.32).
	•	•	CHARTS	Lexical element (see 13.1.4).
•	•	•	charts-element	Single elements of CHARTS may be changed.
	•	•	FILES	Lexical element (see 13.1.8).
•	•	•	files-element	Single elements of FILES may be changed.
	•	•	LISTS	Lexical element (see 13.1.16).
•	•	•	lists-element	Single elements of LISTS may be changed.
	•	•	GRAPHS	Lexical element (see 13.1.10).
•	•	•	graphs-element	Single elements of GRAPHS may be changed.
	•	•	GRIDS	Lexical element (see 13.1.12).
•	•	•	grids-element	Single elements of GRIDS may be changed.
	•	•	MENUS	Lexical element (see 13.1.19).
•	•	•	menus-element	Single elements of MENUS may be changed.
	•	•	METHODS	Lexical element (see 13.1.21).
•	•	•	methods-element	Single elements of METHODS may be changed.
	•	•	PLUGINS	Lexical element (see 13.1.25).
•	•	•	plugins-element	Single elements of PLUGIN may be changed.

15.5 BLOCK_B

Lexical structure

```
BLOCK identifier
{
  [REDEFINE [NUMBER | TYPE]]*
}
```

The redefinition rules of BLOCK_B are specified in Table 208 and Table 212.

Table 212 – Redefinition rules for BLOCK_B attributes

A	D	R	Attribute	Description
		•	NUMBER	Lexical element (see 13.2.2).
		•	TYPE	Lexical element (see 13.2.3).

15.6 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 208 and Table 213.

Table 213 – Redefinition rules for CHART attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 14.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	CYCLE_TIME	Lexical element (see 11.5.2).
	•	•	HEIGHT	Lexical element (see 14.7).
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	LENGTH	Lexical element (see 11.5.3).
	•	•	TYPE	Lexical element (see 11.5.4).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	WIDTH	Lexical element (see 14.19).

15.7 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 208 and Table 214.

Table 214 – Redefinition rules for COLLECTION attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 14.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	PRIVATE	Lexical element (see 14.13).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).

15.8 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 208 and Table 215.

Table 215 – Redefinition rules for COMMAND attributes

A	D	R	Attribute	Description
			OPERATION	Lexical element (see 10.1.8).
•	•	•	TRANSACTION	Lexical element (see 10.1.11).
	•	•	INDEX	Lexical element (see 10.1.6).
			BLOCK_B	Lexical element (see 10.1.4).
			NUMBER	Lexical element (see 10.1.7).
	•	•	SLOT	Lexical element (see 10.1.9).
	•	•	SUB_SLOT	Lexical element (see 10.1.10).
	•	•	API	Lexical element (see 10.1.3).
	•	•	HEADER	Lexical element (see 10.1.5).
	•	•	RESPONSE_CODES	Lexical element (see 14.15).
•	•	•	response-codes-list-element	Single list elements of RESPONSE_CODES may be changed.

15.9 COMPONENT

Lexical structure

```

COMPONENT identifier
{
  [[DELETE | REDEFINE] [LABEL | BYTE_ORDER | CAN_DELETE |
    CHECK_CONFIGURATION | CLASSIFICATION | CONNECTION_POINT | HELP |
    COMPONENT_PARENT | COMPONENT_PATH | COMPONENT_RELATIONS |
    DETECT | EDD | INITIAL_VALUES | PRODUCT_URI | PROTOCOL | REDUNDANCY |
    SCAN | SCAN_LIST | SUPPLIED_INTERFACE]]*
}

```

The redefinition rules of COMPONENT are specified in Table 208 and Table 216.

Table 216 – Redefinition rules for COMPONENT attributes

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 14.9).
	•	•	BYTE_ORDER	Lexical element (see 13.6.2).
	•	•	CAN_DELETE	Lexical element (see 13.6.3).
	•	•	CHECK_CONFIGURATION	Lexical element (see 13.6.4).
	•	•	CLASSIFICATION	Lexical element (see 14.1).
	•	•	COMPONENT_PARENT	Lexical element (see 14.2).
	•	•	COMPONENT_PATH	Lexical element (see 14.3).
	•	•	COMPONENT_RELATIONS	Lexical element (see 13.6.5).
	•	•	CONNECTION_POINT	Lexical element (see 13.6.6).
	•	•	DETECT	Lexical element (see 13.6.7).
	•	•	EDD	Lexical element (see 13.6.8).
	•	•	HELP	Lexical element (see 14.8).
	•	•	INITIAL_VALUES	Lexical element (see 13.6.9).
	•	•	PRODUCT_URI	Lexical element (see 13.6.13).
	•	•	PROTOCOL	Lexical element (see 14.14).
	•	•	REDUNDANCY	Lexical element (see 13.6.10).
	•	•	SCAN	Lexical element (see 13.6.11).
	•	•	SCAN_LIST	Lexical element (see 13.6.12).
	•	•	SUPPLIED_INTERFACE	Lexical element (see 14.16).

15.10 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 208 and Table 217.

Table 217 – Redefinition rules for COMPONENT_FOLDER attributes

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 14.9).
	•	•	CLASSIFICATION	Lexical element (see 14.1).
	•	•	COMPONENT_PARENT	Lexical element (see 14.2).
	•	•	COMPONENT_PATH	Lexical element (see 14.3).
	•	•	HELP	Lexical element (see 14.8).
	•	•	PROTOCOL	Lexical element (see 14.14).

15.11 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 208 and Table 218.

Table 218 – Redefinition rules for COMPONENT_REFERENCE attributes

A	D	R	Attribute	Description
	•	•	CLASSIFICATION	Lexical element (see 14.1).
	•	•	COMPONENT_PARENT	Lexical element (see 14.2).
	•	•	COMPONENT_PATH	Lexical element (see 14.3).
	•	•	DEVICE_REVISION	Lexical element (see 8.3).
	•	•	DEVICE_TYPE	Lexical element (see 8.4).
	•	•	MANUFACTURER	Lexical element (see 8.8).
	•	•	PROTOCOL	Lexical element (see 14.14).

15.12 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 208 and Table 219.

Table 219 – Redefinition rules for COMPONENT_RELATION attributes

A	D	R	Attribute	Description
		•	COMPONENTS	Lexical element (see 13.9.2).
•	•	•	components-list-element	Single list elements of COMPONENTS may be changed.
	•	•	AUTO_CREATE	Lexical element (see 13.9.2).
	•	•	FILTER	Lexical element (see 13.9.2).
	•	•	REQUIRED_RANGES	Lexical element (see 13.9.2).
		•	LABEL	Lexical element (see 14.9).
		•	RELATION_TYPE	Lexical element (see 13.9.3).
	•	•	ADDRESSING	Lexical element (see 13.9.4).
	•	•	HELP	Lexical element (see 14.8).
	•	•	MAXIMUM_NUMBER	Lexical element (see 13.9.5).
	•	•	MINIMUM_NUMBER	Lexical element (see 13.9.6).
	•	•	REQUIRED_INTERFACE	Lexical element (see 13.9.7).
	•	•	SUPPLIED_INTERFACE	Lexical element (see 14.16).

15.13 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 208 and Table 220.

Table 220 – Redefinition rules for EDIT_DISPLAY attributes

A	D	R	Attribute	Description
		•	EDIT_ITEMS	Lexical element (see 11.2.2).
		•	LABEL	Lexical element (see 14.9).
	•	•	DISPLAY_ITEMS	Lexical element (see 11.2.3).
	•	•	POST_EDIT_ACTIONS	Lexical element (see 11.2.4).
	•	•	PRE_EDIT_ACTIONS	Lexical element (see 11.2.5).

15.14 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 208 and Table 221.

Table 221 – Redefinition rules for FILE attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 14.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	HANDLING	Lexical element (see 14.6).
	•	•	HELP	Lexical element (see 14.8).
	•	•	IDENTITY	Lexical element (see 9.6.2).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	SHARED	Lexical element (see 9.6.4).
	•	•	ON_UPDATE_ACTIONS	Lexical element (see 9.6.3).

15.15 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 208 and Table 222.

Table 222 – Redefinition rules for GRAPH attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 14.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	CYCLE_TIME	Lexical element (see 11.8.2).
	•	•	HEIGHT	Lexical element (see 14.7).
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	WIDTH	Lexical element (see 14.19).
	•	•	X_AXIS	Lexical element (see 11.8.3).

15.16 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 208 and Table 223.

Table 223 – Redefinition rules for GRID attributes

A	D	R	Attribute	Description
		•	VECTORS	Lexical element (see 11.4.3).
	•	•	HANDLING	Lexical element (see 14.6).
	•	•	HEIGHT	Lexical element (see 14.7).
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	ORIENTATION	Lexical element (see 11.4.2).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	WIDTH	Lexical element (see 14.19).

15.17 IMAGE**Lexical structure**

```

IMAGE identifier
{
  [ [DELETE | REDEFINE] [PATH | HELP | LABEL | LINK | VALIDITY |
    VISIBILITY] ] *
}

```

The redefinition rules of GRAPH are specified in Table 208 and Table 224.

Table 224 – Redefinition rules for IMAGE attributes

A	D	R	Attribute	Description
		•	PATH	Lexical element (see 11.3.3).
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	LINK	Lexical element (see 11.3.2).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).

15.18 INTERFACE**Lexical structure**

```

INTERFACE identifier
{
  [ [DELETE | REDEFINE] [DECLARATION | LABEL | HELP] ] *
}

```

The redefinition rules of INTERFACE are specified in Table 208 and Table 225.

Table 225 – Redefinition rules for INTERFACE attributes

A	D	R	Attribute	Description
		•	DECLARATION	Lexical element (see 13.10.2).
		•	LABEL	Lexical element (see 14.9).
	•	•	HELP	Lexical element (see 14.8).

15.19 LIST

Lexical structure

LIST *identifier*

```
{
  [[DELETE | REDEFINE] [TYPE | CAPACITY | COUNT | HELP | LABEL | PRIVATE |
    VALIDITY | VISIBILITY]]*
}
```

The redefinition rules of LIST are specified in Table 208 and Table 226.

Table 226 – Redefinition rules for LIST attributes

A	D	R	Attribute	Description
		•	TYPE	Lexical element (see 9.4.4).
	•	•	CAPACITY	Lexical element (see 9.4.2).
	•	•	COUNT	Lexical element (see 9.4.3).
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	PRIVATE	Lexical element (see 14.13).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).

15.20 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 208 and Table 227.

Table 227 – Redefinition rules for MENU attributes

A	D	R	Attribute	Description
		•	ITEMS	Lexical element (see 11.1.5).
		•	LABEL	Lexical element (see 14.9).
	•	•	ACCESS	Lexical element (see 11.1.2).
	•	•	HELP	Lexical element (see 14.8).
	•	•	EXIT_ACTIONS	Lexical element (see 11.1.3).
	•	•	INIT_ACTIONS	Lexical element (see 11.1.4).
	•	•	POST_EDIT_ACTIONS	Lexical element (see 11.1.6).
	•	•	POST_READ_ACTIONS	Lexical element (see 11.1.7).
	•	•	POST_WRITE_ACTIONS	Lexical element (see 11.1.8).
	•	•	PRE_EDIT_ACTIONS	Lexical element (see 11.1.9).
	•	•	PRE_READ_ACTIONS	Lexical element (see 11.1.10).
	•	•	PRE_WRITE_ACTIONS	Lexical element (see 11.1.11).
	•	•	STYLE	Lexical element (see 11.1.12).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).

15.21 METHOD**Lexical structure****METHOD** *identifier*

```

{
  [[DELETE | REDEFINE] [DEFINITION | ACCESS | CLASS | LABEL | HELP |
    PRIVATE | VALIDITY | VISIBILITY]] *
}

```

The redefinition rules of **METHOD** are specified in Table 208 and Table 228.

Table 228 – Redefinition rules for METHOD attributes

A	D	R	Attribute	Description
		•	DEFINITION	Lexical element (see 14.4).
	•	•	ACCESS	Lexical element (see 12.2).
		•	CLASS	Lexical element (see 12.3).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	HELP	Lexical element (see 14.8).
			TYPE	Lexical element (see 12.4). TYPE shall not be redefined.
	•	•	PRIVATE	Lexical element (see 14.13).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).

15.22 PLUGIN

Lexical structure

```

PLUGIN identifier
{
  [[DELETE | REDEFINE] [LABEL | HELP | UUID | VALIDITY | VISIBILITY]] *
}

```

The redefinition rules of **PLUGIN** are specified in Table 208 and Table 229.

Table 229 – Redefinition rules for **PLUGIN attributes**

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 14.9).
	•	•	HELP	Lexical element (see 14.8).
		•	UUID	Lexical element (see 11.10.2).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).

15.23 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 208 and Table 230.

Table 230 – Redefinition rules for **RECORD attributes**

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 14.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
		•	LABEL	Lexical element (see 14.9).
	•	•	HELP	Lexical element (see 14.8).
	•	•	PRIVATE	Lexical element (see 14.13).
	•	•	RESPONSE_CODES	Lexical element (see 14.15).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	WRITE_MODE	Lexical element (see 14.20).

15.24 REFERENCE_ARRAY

Lexical structure

Communication profiles: FF, ISA100

reference-array-keyword := **ITEM_ARRAY**

Communication profiles: CS, GPE, HART, PB, PN

reference-array-keyword := **ARRAY**

reference-array-keyword identifier

```
{
  [[ADD | DELETE | REDEFINE] [ELEMENTS | elements-list-element | HELP |
    LABEL | PRIVATE | VALIDITY | VISIBILITY]]*
}
```

The redefinition rules of **REFERENCE_ARRAY** are specified in Table 208 and Table 231.

Table 231 – Redefinition rules for **REFERENCE_ARRAY attributes**

A	D	R	Attribute	Description
		•	ELEMENTS	Lexical element (see 9.2.2).
•	•	•	<i>elements-list-element</i>	Single list elements of ELEMENTS may be changed.
	•	•	LABEL	Lexical element (see 14.9).
	•	•	HELP	Lexical element (see 14.8).
	•	•	PRIVATE	Lexical element (see 14.13).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).

15.25 **RESPONSE_CODES**

Lexical structure

RESPONSE_CODES *identifier*

```
{
  [[ADD | DELETE | REDEFINE] response-code-list-element]*
}
```

The redefinition rules of **RESPONSE_CODES** are specified in Table 208 and Table 232.

Table 232 – Redefinition rules for **RESPONSE_CODES attributes**

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

15.26 **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 208 and Table 233.

Table 233 – Redefinition rules for SOURCE attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 14.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	EMPHASIS	Lexical element (see 14.5).
	•	•	EXIT ACTIONS	Lexical element (see 11.6.2).
	•	•	HELP	Lexical element (see 14.8).
	•	•	INIT_ ACTIONS	Lexical element (see 11.6.3).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	LINE_COLOR	Lexical element (see 14.10).
	•	•	LINE_TYPE	Lexical element (see 14.11).
	•	•	REFRESH_ ACTIONS	Lexical element (see 11.6.4).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	Y_AXIS	Lexical element (see 11.6.5).

15.27 TEMPLATE

Lexical structure

TEMPLATE *identifier*

```
{
  [[DELETE | REDEFINE] [DEFAULT_VALUES | LABEL | HELP, VALIDITY]] *
}
```

The redefinition rules of TEMPLATE are specified in Table 208 and Table 234.

Table 234 – Redefinition rules for TEMPLATE attributes

A	D	R	Attribute	Description
		•	DEFAULT_VALUES	Lexical element (see 13.5.2).
		•	LABEL	Lexical element (see 14.9).
	•	•	HELP	Lexical element (see 14.8).
	•	•	VALIDITY	Lexical element (see 14.17).

15.28 VALUE_ARRAY

Lexical structure

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 208 and Table 235.

Table 235 – Redefinition rules for VALUE_ARRAY attributes

A	D	R	Attribute	Description
		•	LABEL	Lexical element (see 14.9).
		•	NUMBER_OF_ELEMENTS	Lexical element (see 9.3.2).
		•	TYPE	Lexical element (see 9.3.3).
	•	•	HELP	Lexical element (see 14.8).
	•	•	PRIVATE	Lexical element (see 14.13).
	•	•	RESPONSE_CODES	Lexical element (see 14.15).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	WRITE_MODE	Lexical element (see 14.20).

15.29 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 208 and Table 236.

Table 236 – Redefinition rules for VARIABLE attributes

A	D	R	Attribute	Description
		•	CLASS	Lexical element (see 9.1.2).
	•	•	LABEL	Lexical element (see 14.9).
			TYPE	Lexical element (see 9.1.15). TYPE shall not be redefined.
•	•	•	type-list-element	Single enumeration and bit-enumeration list elements may be changed.
	•	•	DEFAULT_VALUE	Lexical element (see 9.1.4).
	•	•	DISPLAY_FORMAT	Lexical element (see 9.1.15.4.2).
	•	•	EDIT_FORMAT	Lexical element (see 9.1.15.4.3).
	•	•	INITIAL_VALUE	Lexical element (see 9.1.5).
	•	•	MAX_VALUE	Lexical element (see 9.1.15.2).
	•	•	MIN_VALUE	Lexical element (see 9.1.15.2).
	•	•	SCALING_FACTOR	Lexical element (see 9.1.15.4.4).
	•	•	TIME_FORMAT	Lexical element (see 9.1.15.5.2).
	•	•	TIME_SCALE	Lexical element (see 9.1.15.5.3).
	•	•	CONSTANT_UNIT	Lexical element (see 9.1.3).
	•	•	HANDLING	Lexical element (see 14.6).
	•	•	HEIGHT	Lexical element (see 14.7).
	•	•	HELP	Lexical element (see 14.8).
	•	•	POST_EDIT_ACTIONS	Lexical element (see 9.1.6).
	•	•	POST_READ_ACTIONS	Lexical element (see 9.1.7).
	•	•	POST_RQSTUPDATE_ACTIONS	Lexical element (see 9.1.8).
	•	•	POST_USERCHANGE_ACTIONS	Lexical element (see 9.1.9).
	•	•	POST_WRITE_ACTIONS	Lexical element (see 9.1.10).
	•	•	PRE_EDIT_ACTIONS	Lexical element (see 9.1.11).
	•	•	PRE_READ_ACTIONS	Lexical element (see 9.1.12).
	•	•	PRE_WRITE_ACTIONS	Lexical element (see 9.1.13).
	•	•	PRIVATE	Lexical element (see 14.13).
	•	•	REFRESH_ACTIONS	Lexical element (see 9.1.14).
	•	•	RESPONSE_CODES	Lexical element (see 14.15).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	WIDTH	Lexical element (see 14.19).
	•	•	WRITE_MODE	Lexical element (see 14.20).

15.30 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 208 and Table 237.

Table 237 – Redefinition rules for VARIABLE_LIST attributes

A	D	R	Attribute	Description
		•	MEMBERS	Lexical element (see 14.12).
•	•	•	members-list-element	Single list elements of MEMBERS may be changed.
	•	•	HELP	Lexical element (see 14.8).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	RESPONSE_CODES	Lexical element (see 14.15).

15.31 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 208 and Table 238.

Table 238 – Redefinition rules for WAVEFORM attributes

A	D	R	Attribute	Description
		•	TYPE	Lexical element (see 11.9.6).
•		•	X_VALUES	Lexical element (see 11.9.6).
•		•	Y_VALUES	Lexical element (see 11.9.6).
	•	•	NUMBER_OF_POINTS	Lexical element (see 11.9.6).
		•	X_INITIAL	Lexical element (see 11.9.6).
		•	X_INCREMENT	Lexical element (see 11.9.6).
	•	•	EMPHASIS	Lexical element (see 14.5).
	•	•	EXIT_ACTIONS	Lexical element (see 11.9.2).
	•	•	HANDLING	Lexical element (see 14.6).
	•	•	HELP	Lexical element (see 14.8).
	•	•	INIT_ACTIONS	Lexical element (see 11.9.3).
	•	•	KEY_POINTS	Lexical element (see 11.9.4).
	•	•	LABEL	Lexical element (see 14.9).
	•	•	LINE_COLOR	Lexical element (see 14.10).
	•	•	LINE_TYPE	Lexical element (see 14.11).
	•	•	REFRESH_ACTIONS	Lexical element (see 11.9.5).
	•	•	VALIDITY	Lexical element (see 14.17).
	•	•	VISIBILITY	Lexical element (see 14.18).
	•	•	Y_AXIS	Lexical element (see 11.9.7).

16 Referencing

16.1 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

identifier

The attribute is specified in Table 239.

Table 239 – Referencing an EDD instance

Attribute	Description
identifier	Identifier of an EDD instance.

16.2 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 240.

Table 240 – Referencing AXIS of a GRAPH. SOURCE, WAVEFORM

Attribute	Description
reference	Reference to a GRAPH instance, a reference to a SOURCE instance, or a reference to a WAVEFORM instance.
X_AXIS	Shall only be used with GRAPH; Y_AXIS used in all other cases.
Y_AXIS	Shall only be used with SOURCE and WAVEFORM; X_AXIS used in all other cases.

16.3 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 241.

Table 241 – Referencing elements of VARIABLE

Attribute	Description
variable-reference	Reference to a VARIABLE instance. The type of the VARIABLE instance shall be BIT_ENUMERATED.
bit-mask	Bit-mask which identifies a single bit within the variable. The bit-mask shall be a constant expression.

16.4 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 242.

Table 242 – Referencing members of CHART

Attribute	Description
chart-reference	Reference to a CHART instance.
member-identifier	Identifier of an item of MEMBERS.

16.5 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 243.

Table 243 – Referencing members of COLLECTION

Attribute	Description
collection-reference	Reference to a COLLECTION instance.
member-identifier	Identifier of an item of MEMBERS.

16.6 Referencing COMPONENT instances

Purpose

This referencing is used to reference an instance that is related to a COMPONENT_RELATION to 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 and the instance does not exist, the METHOD will be aborted. Such a reference may not be used within a conditional expression.

Lexical structure

reference [expression] . identifier

The attributes of component referencing are specified in Table 244.

Table 244 – Referencing a COMPONENT instance

Attribute	Description
reference	Reference to a COMPONENT.
expression	Expression, which shall be evaluated to a non-negative integer that is within the index range of the COMPONENT_RELATION.
identifier	Identifier of a basic construct of the referenced COMPONENT.

16.7 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 245.

Table 245 – Referencing a COMPONENT type

Attribute	Description
reference	reference to a COMPONENT
identifier	identifier of a basic construct of the referenced COMPONENT

16.8 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 246.

Table 246 – Referencing members of FILE

Attribute	Description
file-reference	Reference to a FILE instance.
member-identifier	Identifier of an item of MEMBERS.

16.9 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 247.

Table 247 – Referencing members of GRAPH

Attribute	Description
graph-reference	Reference to a GRAPH instance.
member-identifier	identifier of an item of MEMBERS.

16.10 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 248.

Table 248 – Referencing elements of LIST

Attribute	Description
list-reference	Reference to a LIST instance.
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 Clause 19). Communication profiles: FF The numbering of elements for LIST starts with 1.
FIRST	First element of the LIST.
LAST	Last element of the LIST.

16.11 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 249.

Table 249 – Referencing elements of RECORD

Attribute	Description
record-reference	Reference to a RECORD instance.
member-identifier	Identifier of an item of MEMBERS.

16.12 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 250.

Table 250 – Referencing members of REFERENCE_ARRAY

Attribute	Description
reference-array-reference	Reference to a REFERENCE_ARRAY instance.
expression	Expression which shall be evaluated to a non-negative integer that is within the index range of the REFERENCE_ARRAY; for description of expressions, see Clause 19.

16.13 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 251.

Table 251 – Referencing members of SOURCE

Attribute	Description
source-reference	Reference to a SOURCE instance.
member-identifier	Identifier of an item of MEMBERS.

16.14 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 252.

Table 252 – Referencing elements of VALUE_ARRAY

Attribute	Description
value-array-reference	Reference to a VALUE_ARRAY instance.
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 Clause 19. Communication profiles: FF The numbering of elements for VALUE_ARRAY starts with 1.

16.15 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 253.

Table 253 – Referencing members of VARIABLE_LISTS

Attribute	Description
variable-list-reference	Reference to a VARIABLE_LISTS instance.
member-identifier	Identifier of an item of MEMBERS.

16.16 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 254.

Table 254 – Referencing BLOCK_A CHARACTERISTICS

Attribute	Description
characteristics-identifier	Identifier of the CHARACTERISTICS reference.

16.17 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 255.

Table 255 – Referencing members of BLOCK_A LOCAL_PARAMETER

Attribute	Description
member-identifier	Identifier of an item of LOCAL_PARAMETER.

16.18 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 256.

Table 256 – Referencing members of a BLOCK_A PARAMETERS

Attribute	Description
member-identifier	Identifier of an item of PARAMETERS.

16.19 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 257.

Table 257 – Referencing members of BLOCK_A PARAMETER_LISTS

Attribute	Description
member-identifier	Identifier of an item of PARAMETER_LISTS.

16.20 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 258.

Table 258 – Referencing CHARACTERISTICS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.21 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 259.

Table 259 – Referencing CHARTS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.22 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 260.

Table 260 – Referencing FILES of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.23 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 261.

Table 261 – Referencing GRAPHS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.24 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 262.

Table 262 – Referencing GRIDS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.25 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 263.

Table 263 – Referencing LISTS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.26 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 264.

Table 264 – Referencing LOCAL_PARAMETERS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.27 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 265.

Table 265 – Referencing MENUS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.28 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 266.

Table 266 – Referencing METHODS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.29 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 267.

Table 267 – Referencing PARAMETERS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

16.30 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 268.

Table 268 – Referencing PLUGINS of specific BLOCK_A instance

Attribute	Description
reference	Reference to a BLOCK_A instance.
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.
member-identifier	Identifier of an item of MEMBERS.

17 Strings

17.1 Specifying a string as a string literal

Purpose

A text string can be specified as a string literal (see A.2.3). Text strings can be specified for one or more languages (see A.2.4).

Lexical structure

string

The attribute is specified in Table 269.

Table 269 – String as a string literal

Attribute	Description
string	String literal.

17.2 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 270.

Table 270 – String as a string variable

Attribute	Description
reference	Reference to a VARIABLE instance.

17.3 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 (expr)

The attributes are specified in Table 271.

Table 271 – String as an enumeration value

Attribute	Description
reference	Reference to a VARIABLE instance of TYPE ENUMERATED or BIT_ENUMERATED.
expr	Resolves to an integer corresponding to the 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).

EXAMPLE 3 The following enumeration value string specifies the enumerated string of the ENUMERATED variable units_code associated with the current value of units_code: units_code(units_code)

17.4 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 272.

Table 272 – String as a dictionary reference

Attribute	Description
reference	Reference to one or more strings specified in a dictionary (see Clause 20).

17.5 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 273.

Table 273 – Referencing HELP and LABEL attributes of EDD instances

Attribute	Description
reference	EDD instance which contains the attributes LABEL and/or HELP.
HELP	HELP string of the instance.
LABEL	LABEL string of the instance.

17.6 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-expression [*concatenation-operator* *string-expression*]+

The attributes are specified in Table 274.

Table 274 – String operation

Attribute	Description
string-expression	Either string literals, string or numeric data type variables, string enumeration values, dictionary references or references to string attributes, e.g. HELP or LABEL. The EDD application shall convert evaluated expressions and numeric data type variable to strings (see Clause A.3).
concatenation-operator	The strings shall be concatenated.

18 Conditional expression

18.1 General structure

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.

- IF conditionals, for specifying an attribute that has two alternative definitions.
- SELECT conditionals, for specifying an attribute that has several alternative definitions.
- A primary, unary or binary expression. 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 IF or SELECT conditional.

18.2 IF conditional

Purpose

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.

Lexical structure for the IF conditional

```
IF ( if-expression )
{
    then-clause+
}
[ELSE
{
    else-clause+
}]?
```

The attributes are specified in Table 275.

Table 275 – IF conditional

Attribute	Description
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.
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.
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.

If the

- then-clause,
- else-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.

18.3 SELECT conditional

Purpose

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.

Lexical structure for the SELECT conditional

```
SELECT ( select-expression )
{
  [CASE integer_n: select-clause]+
  [DEFAULT: default-clause]?
}
```

The attributes are specified in Table 276.

Table 276 – SELECT conditional

Attribute	Description
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.
integer_n	A number which identifies the according CASE. If integer matches with integer_n, the according select_clause is used.
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.
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

- 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.

19 Expression

19.1 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.

19.2 Primary expressions

Table 277 below summarizes the primary expressions in the EDDL.

Table 277 – 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 corresponding value of the attribute (see Table 278 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 according value of the attributes (see Table 279 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 according value of the attributes (see Table 281 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 278 – 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 22) shall be used for referencing. If no MIN_VALUE is defined, the data type specific minimum value (see Table B.2) 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 22) shall be used for referencing. If no MAX_VALUE is defined, the data type specific maximum value (see Table B.2) 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 B.2) 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 B.2) should be used in the evaluation.
VARIABLE_STATUS		Communication profiles: HART, FF, ISA100 Expressions containing VARIABLE_STATUS are not supported. 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 279 – 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 280 summarizes the attributes of a BLOB.

Table 280 – 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 281 – 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 282 summarizes the attributes of an ARRAY.

Table 282 – ARRAY attribute values

Attributes	Data type	Description
NUMBER_OF_ELEMENTS	integer	Value which complies with the NUMBER_OF_ELEMENTS of the VALUE_ARRAY instance.

19.3 Unary expressions

A unary expression consists of a numerical operand, an expression, preceded by a unary operator. Table 283 specifies the unary expressions in the EDDL.

Table 283 – 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.

19.4 Binary expressions

19.4.1 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. The following types of binary expressions are specified:

- Multiplicative
- Additive
- Shift
- Relational
- Equality
- Bitwise AND (&)
- Bitwise XOR (^)
- Bitwise OR (|)
- Logical AND (&&)
- Logical OR (||)
- Conditional evaluation

19.4.2 Multiplicative operators

Multiplicative operators specify the multiplication and division of operands. Table 284 specifies the multiplicative operators.

Table 284 – 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.

19.4.3 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 285 specifies the additive operators.

Table 285 – Additive operators

Operator	Description
+	Addition of its operands.
-	Subtraction of the second operand from the first.

19.4.4 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 286 specifies the shift operators.

Table 286 – 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.

19.4.5 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 287 specifies the relational operators.

Table 287 – 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".

19.4.6 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 288 specifies the equality operators.

Table 288 – Equality operators

Operator	Description
==	Tests for the relationship "equals".
!=	Tests for the relationship "does not equal".

19.4.7 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.

19.4.8 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.

19.4.9 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.

19.4.10 Logical AND operator (&&)

The && operator specifies the Boolean AND evaluation of its integer operands. If either operand references a VARIABLE value that cannot be determined (e.g. reading from an online device fails), 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.

19.4.11 Logical OR operator (||)

The || operator specifies the Boolean OR evaluation of its integer operands. If either operand references a VARIABLE value that cannot be determined (e.g. no read command, reading from an online device fails, the device does not respond), 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.

19.4.12 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; }

20 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*] | *string*) ([[*language-code*] [*country-code*]? *string*)]*

The attributes are specified in Table 289.

Table 289 – 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.

For more details about string literals and the usage of language and country codes, see A.2.3 and A.2.4.

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__`

`__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__`

`__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/IEC 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.

A string literal can 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	Support
\'	Single quote.	Yes
\"	Double quote.	Yes
\\	Backslash.	Yes
\f	Form feed.	No. If encountered, EDD application shall use a new line in its character position.
\n	New line.	Yes
\r	Carriage return.	Yes. This is same as \n. If combined \r\n sequence is encountered, it shall be treated as a single newline.
\t	Horizontal tab.	Yes
\v	Vertical tab.	No. If encountered, EDD application shall use a new line in its character position.

Any other escape sequence not listed in Table A.2 shall be ignored by the EDD application.

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	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.	== != < <= >= > =

VARIABLE TYPE	METHOD local data type	Operators
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 []	+ == != [] =
OCTET	unsigned char[]	+ == != [] =
N.a.	time_t	== != =
Key: N.a.: not applicable		

Concatenation of string literals without '+' operator shall be also supported (see A.7.33, "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	COLUMNWIDTH_EQUAL	COLUMNWIDTH_OPTIMIZED
COMM_ERROR	COMMAND	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	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

EDDL keyword	EDDL keyword	EDDL keyword
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
int	INTEGER	INTERFACE
INTERFACES	IS_CONFIG	ITEM_ARRAY
ITEM_ARRAY_ITEMS	ITEM_ARRAYS	ITEMS
KEY_POINTS		
LABEL	LARGE	LAST
LAYOUT_TYPE	LENGTH	LIKE
LINE_COLOR	LINE_TYPE	LINEAR
LINK	LIST	LIST_ITEMS
LISTS	LOAD_TO_APPLICATION	LOAD_TO_DEVICE
LOCAL	LOCAL_A	LOCAL_B
LOCAL_DISPLAY	LOCAL_PARAM	LOCAL_PARAMETERS

EDDL keyword	EDDL keyword	EDDL keyword
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_WIRELESS_ADAPTER	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_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

EDDL keyword	EDDL keyword	EDDL keyword
REQUIRED_RANGES	RESPONSE_CODES	return
REVIEW	ROWBREAK	
SCALING	SCALING_FACTOR	SCAN
SCAN_LIST	SCOPE	SECONDS
SELECT	SELF	SELF_CORRECTING
SEMANTIC_MAP	SEMANTIC_MAP_ITEMS	SEMANTIC_MAPS
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
TV19	TV20	TYPE
UNCORRECTABLE	UNIT	UNIT_ITEMS
UNITS	UNIVERSAL	unsigned
UNSIGNED_INTEGER	UUID	
VALIDITY	VARIABLE	VARIABLE_LIST
VARIABLE_LISTS	VARIABLE_STATUS	VARIABLE_STATUS_CHANGED
VARIABLE_STATUS_INITIAL	VARIABLE_STATUS_INVALID	VARIABLE_STATUS_LOADED
VARIABLE_STATUS_NONE	VARIABLE_STATUS_NOT_ACCEPTED	VARIABLE_STATUS_NOT_CONVERTED

EDDL keyword	EDDL keyword	EDDL keyword
VARIABLE_STATUS_NOT_SUPPORTED	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 Symbols and symbol scope

All EDD items have one or more (non-keyword) identifiers associated with them. These identifiers are called symbols. Examples of symbols include EDD item names and member-names. Symbols are used within an EDD as EDD item references and often maintained over long periods of time by EDD applications that archive historical information about the device instances associated with an EDD. Furthermore, some symbols have special significance and conventions are associated with their use.

Symbols shall be unique within their scope. Symbols defined within a device EDD have scope local to that EDD and its revisions. It can only be referenced within that EDD. However, standard EDDs may be IMPORTED by any EDD and, consequently, the symbols they define may be used by any EDD. They have a global scope and any EDD referencing these "standard symbols" shall IMPORT the EDD defining them and cannot change the semantic associated with them.

A.6 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.7 Formal EDDL syntax

A.7.1 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.7.2 EDD identification information

```

device_description:
    = identification definition_list

```

```

identification:
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' layout_type
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' layout_type
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_profile
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_profile ',' layout_type
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' manufacturer_ext ',' layout_type
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' manufacturer_ext ',' device_revision ','
      DD_revision ',' edd_version ',' edd_profile
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' edd_profile ','

```

```

    layout_type
= manufacturer ',' device_type ',' device_revision ','
  DD_revision ',' edd_version ',' manufacturer_ext
= manufacturer ',' device_type ',' device_revision ','
  DD_revision ',' edd_version ',' manufacturer_ext ','
  layout_type
= manufacturer ',' device_type ',' device_revision ','
  DD_revision ',' edd_profile ',' manufacturer_ext
= manufacturer ',' device_type ',' device_revision ','
  DD_revision ',' edd_profile ',' manufacturer_ext ','
  layout_type
= manufacturer ',' device_type ',' device_revision ','
  DD_revision ',' edd_version ',' edd_profile ','
  manufacturer_ext
= manufacturer ',' device_type ',' device_revision ','
  DD_revision ',' edd_version ',' edd_profile ','
  manufacturer_ext ',' layout_type

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

layout_type:
  = 'COLUMNWIDTH_EQUAL'
  = 'COLUMNWIDTH_OPTIMIZED'

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.7.3 AXIS

axis:

```

= 'AXIS' Identifier '{' axis_attribute_list '}'
= 'AXIS' Identifier '{' '}'

```

axis_attribute_list:

```

= axis_attribute
= axis_attribute_list axis_attribute

```

axis_attribute:

```

= constant_unit           /* 0 */
= help                    /* 0 */
= label                   /* 0 */
= maximum_value           /* 0 */
= minimum_value           /* 0 */
= axis_scaling            /* 0 */

```

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.7.4 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_semantic_map_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_semantic_map_items:
    = 'SEMANTIC_MAP_ITEMS' '{' semantic_map_items_specifier_list '}'

semantic_map_items_specifier_list:
    = semantic_map_items_specifier
    = semantic_map_items_specifier_list ',' semantic_map_items_specifier

semantic_map_items_specifier:
    = semantic_map_items_reference

semantic_map_items_reference:
    = reference_without_call

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.7.5 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.7.6 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'
    = 'SEMANTIC_MAP'

```

```
= 'TEMPLATE'
= 'UNIT'
= 'WAVEFORM'
= 'WRITE_AS_ONE'
```

```
collection_attribute_list:
= collection_attribute
= collection_attribute_list collection_attribute
```

```
collection_attribute:
= collection_members          /* M */
= help                        /* O */
= label                        /* O */
= private                     /* O */
= validity                     /* O */
= visibility                   /* O */
```

```
collection_members:
= 'MEMBERS' '{' '}'
= 'MEMBERS' '{' members_specifier_list '}'
```

A.7.7 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 */
= 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                    /* M */
= reply                      /* M */
= 'RESPONSE_CODES' '(' response_code_reference ')' /* O */
```

```

    = post_rqstreceive_actions                                /* 0 */

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
    = reference_without_call
    = reference_without_call '<' Integer '>'
    = reference_without_call '(' data_items_qualifiers ')'
    = reference_without_call '<' Integer '>' '(' data_items_qualifiers ')'

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.7.8 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 */
    = 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_literal ';'

product_uri:
= 'PRODUCT_URI' string_literal ';'

protocol:
= 'PROTOCOL' protocol_specifier ';'

protocol_specifier:
= 'PROFIBUS_DP'
= 'PROFIBUS_PA'
= 'PROFINET'
= 'HART'
= 'FF'
= 'ISA100_WCI'
= 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.7.9 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.7.10 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.7.11 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.7.12 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:
  = reference_without_call

```

A.7.13 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.7.14 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.7.15 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.7.16 IMAGE

```

image:
    = 'IMAGE' Identifier '{' image_attribute_list '}'

image_attribute_list:
    = image_attribute
    = image_attribute_list image_attribute

image_attribute:
    = image_path                          /* M */
    = label                                /* O */
    = help                                 /* O */
    = image_link                           /* O */
    = validity                             /* O */
    = visibility                           /* 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.7.17 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.7.18 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.7.19 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'
    = 'SEMANTIC_MAPS'
    = '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
= semantic_map_redefinition
= source_redefinition
= template_redefinition
= unit_relation_redefinition
= variable_redefinition
= variable_list_redefinition
= wao_relation_redefinition
= waveform_redefinition

```

```

filename:
    = string_literal

```

A.7.20 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.7.21 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'
= String

```

```

    = dictionary_reference

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'
    = 'NO_LABEL'

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.7.22 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'
= 'BACKGROUND'
= '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.7.23 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.7.24 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' '{' 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.7.25 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.7.26 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.7.27 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.7.28 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.7.29 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.7.30 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          /* O */
    = post_read_actions          /* O */
    = post_refresh_actions       /* O */
    = post_update_actions        /* O */
    = post_userchange_actions    /* O */
    = post_write_actions         /* O */
    = pre_edit_actions           /* O */
    = pre_read_actions           /* O */
    = pre_write_actions          /* O */
    = private                    /* O */
    = refresh_actions            /* O */
    = response_codes             /* O */
    = validity                   /* O */
    = visibility                 /* O */
    = width                     /* O */
    = write_mode                /* O */

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'
= 'CONSTANT'
= 'CONTAINED'
= 'CORRECTION'
= 'DEVICE'
= 'DIAGNOSTIC'
= 'DIGITAL_INPUT'
= 'DIGITAL_OUTPUT'
= 'DISCRETE'
= 'DISCRETE_INPUT'
= 'DISCRETE_OUTPUT'
= 'DYNAMIC'
= 'FREQUENCY'
= 'FREQUENCY_INPUT'
= 'FREQUENCY_OUTPUT'
= 'HART'
= 'INPUT'
= 'IS_CONFIG'
= 'LOCAL'
= 'LOCAL_A'
= 'LOCAL_B'
= 'LOCAL_DISPLAY'
= 'MODE'
= 'OPERATE'
= 'OPTIONAL'
= 'OUTPUT'
= 'SERVICE'
= 'SPECIALIST'
= 'TEMPORARY'
= 'TUNE'
```

type:

```
= 'TYPE' type_specifier
```

type_specifier:

```
= arithmetic_type
= enumerated_type
= index_type
= string_type
= binary_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 ')'
  = AOOn '(' 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
= 'EUC' string_size string_options
= 'PACKED_ASCII' string_size string_options
= 'PASSWORD' string_size string_options
= 'VISIBLE' 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 ';'

binary_type:
    = 'BITSTRING' binary_size binary_bitstring_options
    = 'OCTET' binary_size binary_octet_options

binary_size:
    = '(' Integer ')'

binary_bitstring_options:
    = ';'
    = '{' binary_bitstring_option_list '}'

binary_bitstring_option_list:
    = binary_bitstring_option
    = binary_bitstring_option_list binary_bitstring_option

binary_bitstring_option:
    = binary_bitstring_default_value
    = binary_bitstring_initial_value
    = display_format
    = edit_format

binary_bitstring_default_value:
    = 'DEFAULT_VALUE' octet_values_specifier ';'

binary_bitstring_initial_value:
    = 'INITIAL_VALUE' octet_values ';'

binary_octet_options:
    = ';'
    = '{' binary_octet_option_list '}'

binary_octet_option_list:
    = binary_octet_option
    = binary_octet_option_list binary_octet_option

binary_octet_option:
    = binary_octet_default_value
    = binary_octet_initial_value

```

```

    = display_format
    = edit_format

binary_octet_default_value:
    = string_default_value
    = 'DEFAULT_VALUE' octet_values_specifier ';'

binary_octet_initial_value:
    = string_initial_value
    = 'INITIAL_VALUE' octet_values ';'

octet_values_specifier:
    = octet_values ';'
    = 'IF' '(' expr ')' '{' octet_values_specifier '}'
    = 'IF' '(' expr ')' '{' octet_values_specifier '}'
    = 'ELSE' '{' octet_values_specifier '}'
    = 'SELECT' '(' expr ')' '{' octet_values_selection_list '}'

octet_values_selection_list:
    = octet_values_selection
    = octet_values_selection_list octet_values_selection

octet_values_selection:
    = 'CASE' expr ':' octet_values_specifier
    = 'DEFAULT' ':' octet_values_specifier

octet_values:
    = '{' octet_list '}'

octet_list:
    = octet_element
    = octet_list ',' octet_element

octet_element:
    = Integer

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.7.31 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.7.32 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.7.33 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 ';'
    = '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 ')'
    = enumerated_variable_reference '(' expr ')'
    = dictionary_reference

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.7.34 Expression

```
primary_expr:
```

```
= Integer
= Real
= 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_expr  
= '--' unary_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

assignment_expr:
    = conditional_expr
    = unary_expr assignment_operator assignment_expr

assignment_operator:
    = '='
    = '*='
    = '/='
    = '%='
    = '+='
    = '-='
    = '>>='
    = '<<='
    = '&='
    = '^='
    = '|='

expr:
    = assignment_expr
    = expr ',' assignment_expr

```

A.7.35 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:
    = 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.7.36 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_semantic_map_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_semantic_map_items_redefinition:
    = 'DELETE' 'SEMANTIC_MAP_ITEMS' ';'
    = 'REDEFINE' block_a_semantic_map_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
```

```

= '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
= 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
= private_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
    = method_class_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

method_class_redefinition:
    = 'REDEFINE' method_class

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 '}'

semantic_map_redefinition:
    = 'DELETE' 'SEMANTIC_MAP' Identifier ';'
    = 'REDEFINE' 'SEMANTIC_MAP' Identifier
      '{' semantic_id_list ':' semantic_item_list '}'

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' binary_type
    = 'TYPE' binary_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 '}'
    = 'EUC' 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

```

```

binary_type_redefinition:
    = 'BITSTRING' binary_size '{' binary_bitstring_option_redefinition_list
    '}'

    = 'OCTET' binary_size '{' binary_octet_option_redefinition_list '}'

binary_bitstring_option_redefinition_list:
    = binary_bitstring_option_redefinition
    = binary_bitstring_option_redefinition_list
binary_bitstring_option_redefinition

binary_bitstring_option_redefinition:
    = binary_bitstring_default_value_redefinition
    = binary_bitstring_initial_value_redefinition
    = display_format_redefinition
    = edit_format_redefinition

binary_bitstring_default_value_redefinition:
    = 'DELETE' 'DEFAULT_VALUE' ';'
    = 'REDEFINE' binary_bitstring_default_value

binary_bitstring_initial_value_redefinition:
    = 'DELETE' 'INITIAL_VALUE' ';'
    = 'REDEFINE' binary_bitstring_initial_value

binary_octet_option_redefinition_list:
    = binary_octet_option_redefinition
    = binary_octet_option_redefinition_list
binary_octet_option_redefinition

binary_octet_option_redefinition:
    = binary_octet_default_value_redefinition
    = binary_octet_initial_value_redefinition
    = display_format_redefinition
    = edit_format_redefinition

binary_octet_default_value_redefinition:
    = 'DELETE' 'DEFAULT_VALUE' ';'
    = 'REDEFINE' binary_octet_default_value

binary_octet_initial_value_redefinition:
    = 'DELETE' 'INITIAL_VALUE' ';'
    = 'REDEFINE' binary_octet_initial_value

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.7.37 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

semantic_map_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.7.38 PLUGIN

```

plugin:
    = 'PLUGIN' Identifier '{' plugin_attribute_list '}'

plugin_attribute_list:
    = plugin_attribute
    = plugin_attribute_list plugin_attribute

plugin_attribute:
    = label
    = help
    = uuid

```

```

/* M */
/* O */
/* M */

```

```
= validity          /* O */
= visibility         /* O */
```

```
plugin_uuid:
    = 'UUID' uuid ';'


```

A.7.39 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.7.40 SEMANTIC_MAP

```
semantic_map:
    = 'SEMANTIC_MAP' Identifier
    = '{' semantic_id_list ':' semantic_item_list '}'


```

```
semantic_id_list:
    = semantic_id
    = semantic_id_list ',' semantic_id


```

```
semantic_id:
    string_literal


```

```
semantic_item_list:
    = semantic_item_specifier
    = semantic_item_list ',' semantic_item_specifier


```

```
semantic_item_specifier:
    = reference
    = reference '{' semantic_item_option_list '}'


```

```
semantic_item_option_list:
    = semantic_item_option
    = semantic_item_option_list semantic_item_option


```

```
semantic_item_option:
    = semantic_item_constant_unit
    = semantic_item_enumerator_list


```

```
semantic_item_constant_unit:
    = 'CONSTANT_UNIT' string_literal ';'


```

```
semantic_item_enumerator_list:
    = semantic_item_enumerator
    = semantic_item_enumerator_list , semantic_item_enumerator


```

```
semantic_item_enumerator:
    = '{' arithmetic_value ',' semantic_id_enum '}'


```

```
semantic_id_enum:
    string_literal


```

A.8 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.6: Integer, Identifier
    Already defined in A.7.33: string_literal
*/
```

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020

Annex B (normative)

Data types

B.1 METHOD DEFINITION data types

Table B.1 specifies the METHOD DEFINITION data types.

Table B.1 – 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	$-3,402\ 823\ 466 \times 10^{38}$	$+3,402\ 823\ 466 \times 10^{38}$	—
double	$-1,797\ 693\ 134\ 862\ 315\ 7 \times 10^{308}$	$+1,797\ 693\ 134\ 862\ 315\ 7 \times 10^{308}$	—
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.

The DD_STRING type is actually a string object and its internal structure and means used to represent characters are unspecified. The storage space required for a DD_STRING is not specified but a DD_STRING shall support at least 1 024 characters.

B.2 VARIABLE TYPE data types

B.2.1 General

Table B.2 specifies the VARIABLE TYPE data types. The specified size in brackets of the VARIABLE TYPES ensures that an overflow or underflow cannot occur.

Table B.2 – VARIABLE TYPEs

VARIABLE TYPEs	Minimum value	Maximum value	Initial value
BOOLEAN	See B.2.8	See B.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 B.2.2	See B.2.2	1-Jan-1900
DATE_AND_TIME	See B.2.3	See B.2.3	1-Jan-1900 00:00:00
DURATION	See B.2.4	See B.2.4	0 days, 0 hours, 0 minutes, 0 seconds
TIME	See B.2.5	See B.2.5	01-Jan-1984 00:00:00
TIME_VALUE(4)	See B.2.6	See B.2.6	00:00:00
TIME_VALUE(8)	See B.2.6	See B.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. Communication profiles: HART Each character set to question mark (?).
BITSTRING(size)	0	1 per bit	Each bit set to 0.
EUC(size)	0	+255 per octet	Each character set to space.
OCTET(size)	0	+255 per octet	Each character set to space.
PACKED_ASCII(size)	See B.2.7	See B.2.7	Each character set to question mark (?).
PASSWORD(size)	0	+255 per character	Each character set to space. Communication profiles: HART Each character set to question mark (?).
VISIBLE(size)	0	+127 per character	Each character set to space.
Key:			
N.a.: not applicable			
NOTE The initial value can depend on the communication profile.			

B.2.2 Coding of data DATE

Table B.3 specifies the DATE coding.

Table B.3 – 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)

B.2.3 Coding of data DATE_AND_TIME

Table B.4 specifies the DATE_AND_TIME coding.

Table B.4 – 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) Communication profiles: FF The actual year is coded without century, i.e. 0...99 years.

R means reserved for future use; S = 0 indicates standard time; S = 1 indicates summer time.

B.2.4 Coding of data DURATION

Table B.5 and Table B.6 specifies the DURATION coding.

Communication profiles: CS, FF, GPE, PB, PN

Table B.5 – 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 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
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Communication profiles: ISA100

Table B.6 – DURATION coding for ISA100

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}	Number of seconds
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^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	Fractional seconds
6	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	

B.2.5 Coding of data TIME

Table B.7 and Table B.8 specifies the TIME coding.

Communication profiles: CS, FF, GPE, PB, PN

Table B.7 – 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 1984-01-01
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Communication profiles: ISA100

Table B.8 – TIME coding for ISA100

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}	Number of seconds since 1958-01-01
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^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	Fractional seconds
6	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	

B.2.6 Coding of data TIME_VALUE

Table B.9 specifies the four-octet TIME_VALUE coding.

Table B.9 – 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	Communication profiles: CS, GPE, HART, PB, PN
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	Time difference or time of day since midnight as positive number of 1/32 ms.
4	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	Communication profiles: ISA100 Number of seconds since 1958-01-01

Table B.10 specifies the eight octet TIME_VALUE coding.

Table B.10 – 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 1 January 1972.
2	2 ⁵⁵	2 ⁵⁴	2 ⁵³	2 ⁵²	2 ⁵¹	2 ⁵⁰	2 ⁴⁹	2 ⁴⁸	
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	

B.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 Bit #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 B.11 shows the coding of the PACKED_ASCII characters.

Table B.11 – 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

B.2.8 Coding of BOOLEAN

A BOOLEAN value is a one-byte unsigned integer. Table B.12 specifies the BOOLEAN coding.

Table B.12 – BOOLEAN coding

Value	Code	Description
FALSE	0	
TRUE	Communication profiles: CS, GPE, HART, PB, PN any other value Communication profiles: FF, ISA100 0xFF	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.

B.2.9 Coding of BIT_ENUMERATED

A BIT_ENUMERATED value is an unsigned integer defined as a set of bit flags. It can be 1, 2, 3, 4, 5, 6, 7, or 8 octets.

Table B.13 and Table B.14 specify the BIT_ENUMERATED bit number coding.

Communication profiles: CS, GPE, HART, PB, PN, ISA100

Table B.13 – BIT_ENUMERATED coding

Octet	Bit							
	8	7	6	5	4	3	2	1
1	7	6	5	4	3	2	1	0
2	15	14	13	12	11	10	9	8
...	etc.							
n								

Communication profiles: FF

Table B.14 – BIT_ENUMERATED coding for FF

Octet	Bit							
	8	7	6	5	4	3	2	1
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
...	etc.							
n								

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020

Annex C (informative)

EDD example

C.1 EDD example of a temperature transmitter

The following EDD example describes a temperature transmitter which is composed out 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 Clause 7.
- 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	Pt100		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	chdwelskfrgwtb		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 visualized 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 61804-3:2020

Bibliography

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

IEC 61360 (all parts), *Standard data element types with associated classification scheme*

IEC 61499-1:2012, *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/IEEE 8802-3*

IEC 61987 (all parts), *Industrial-process measurement and control – Data structures and elements in process equipment catalogues*

ISO/IEC 2022, *Information technology – Character code structure and extension techniques*

ISO/IEC 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*

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020

SOMMAIRE

AVANT-PROPOS	413
INTRODUCTION.....	416
1 Domaine d'application	417
2 Références normatives	417
3 Termes, définitions, termes abrégés et acronymes	419
3.1 Termes et définitions	419
3.2 Termes abrégés et acronymes	420
4 Déclaration de conformité.....	421
5 Modèle EDD et EDDL	421
5.1 Présentation des EDD et du langage EDDL	421
5.2 Architecture EDD	422
5.3 Concepts d'EDD.....	422
5.4 Principes du processus de développement d'EDD.....	422
5.4.1 Généralités	422
5.4.2 Génération du code source de l'EDD	422
5.4.3 Prétraitement de l'EDD	423
5.4.4 Compilation de l'EDD.....	423
5.5 Interrelations entre la structure lexicale et les définitions formelles	423
5.6 Builtins.....	424
5.7 Profils	424
6 Conventions	424
6.1 Généralités	424
6.2 Convention de description des constructions de base	424
6.3 Conventions pour la structure lexicale.....	425
7 Vue d'ensemble	427
7.1 Caractéristiques du langage EDDL	427
7.2 Représentation de syntaxe.....	427
7.3 Eléments de langage d'une EDD.....	427
7.4 Eléments de construction de base.....	427
7.4.1 Généralités	427
7.4.2 AXIS	428
7.4.3 BLOB.....	428
7.4.4 BLOCK_A	428
7.4.5 BLOCK_B	428
7.4.6 CHART	429
7.4.7 COLLECTION	429
7.4.8 COMMAND	429
7.4.9 COMPONENT.....	430
7.4.10 COMPONENT_FOLDER	430
7.4.11 COMPONENT_REFERENCE	431
7.4.12 COMPONENT_RELATION.....	431
7.4.13 EDIT_DISPLAY	431
7.4.14 FILE	432
7.4.15 GRAPH.....	432
7.4.16 GRID	432
7.4.17 IMAGE.....	432

7.4.18	IMPORT	433
7.4.19	INTERFACE	433
7.4.20	LIKE	433
7.4.21	LIST	433
7.4.22	MENU	434
7.4.23	METHOD	434
7.4.24	PLUGIN	434
7.4.25	RECORD	435
7.4.26	REFERENCE_ARRAY	435
7.4.27	Relations	435
7.4.28	RESPONSE_CODES	436
7.4.29	SOURCE	437
7.4.30	TEMPLATE	437
7.4.31	VALUE_ARRAY	437
7.4.32	VARIABLE	437
7.4.33	VARIABLE_LIST	438
7.4.34	WAVEFORM	438
7.5	Attributs communs	439
7.6	Eléments spéciaux	439
7.7	Règles pour les instances	439
7.8	Règles pour une liste d'attributs VARIABLE	440
8	Informations d'identification d'une EDD	440
8.1	Structure générale	440
8.2	DD_REVISION	441
8.3	DEVICE_REVISION	441
8.4	DEVICE_TYPE	442
8.5	EDD_PROFILE	442
8.6	EDD_VERSION	443
8.7	LAYOUT_TYPE	443
8.8	MANUFACTURER	444
8.9	MANUFACTURER_EXT	445
9	Modélisation de données	445
9.1	VARIABLE	445
9.1.1	Structure générale	445
9.1.2	CLASS	446
9.1.3	CONSTANT_UNIT	450
9.1.4	DEFAULT_VALUE	451
9.1.5	INITIAL_VALUE	451
9.1.6	POST_EDIT_ACTIONS	452
9.1.7	POST_READ_ACTIONS	452
9.1.8	POST_RQSTUPDATE_ACTIONS	453
9.1.9	POST_USERCHANGE_ACTIONS	453
9.1.10	POST_WRITE_ACTIONS	454
9.1.11	PRE_EDIT_ACTIONS	454
9.1.12	PRE_READ_ACTIONS	455
9.1.13	PRE_WRITE_ACTIONS	455
9.1.14	REFRESH_ACTIONS	455
9.1.15	TYPE	456
9.2	REFERENCE_ARRAY	477

9.2.1	Structure générale	477
9.2.2	ELEMENTS	479
9.2.3	item-type	479
9.3	VALUE_ARRAY	479
9.3.1	Structure générale	479
9.3.2	NUMBER_OF_ELEMENTS	480
9.3.3	TYPE	481
9.4	LIST	482
9.4.1	Structure générale	482
9.4.2	CAPACITY	483
9.4.3	COUNT	483
9.4.4	TYPE	484
9.5	COLLECTION	485
9.5.1	Structure générale	485
9.5.2	item-type	486
9.6	FILE	487
9.6.1	Structure générale	487
9.6.2	IDENTITY	488
9.6.3	ON_UPDATE_ACTIONS	489
9.6.4	SHARED	489
9.7	Relations	490
9.7.1	REFRESH	490
9.7.2	SEMANTIC_MAP	491
9.7.3	UNIT	493
9.7.4	WRITE_AS_ONE	494
9.8	BLOB	495
9.8.1	Structure générale	495
9.8.2	IDENTITY	496
10	Communication	497
10.1	COMMAND	497
10.1.1	Structure générale	497
10.1.2	Adressage de commande	498
10.1.3	API	499
10.1.4	BLOCK_B	499
10.1.5	HEADER	500
10.1.6	INDEX	500
10.1.7	NUMBER	501
10.1.8	OPERATION	501
10.1.9	SLOT	502
10.1.10	SUB_SLOT	503
10.1.11	TRANSACTION	503
10.2	RECORD	510
10.3	RESPONSE_CODES	511
10.4	VARIABLE_LIST	512
11	Interface utilisateur	513
11.1	MENU	513
11.1.1	Structure générale	513
11.1.2	ACCESS	514
11.1.3	EXIT_ACTIONS	515

11.1.4	INIT_ACTIONS	515
11.1.5	ITEMS	515
11.1.6	POST_EDIT_ACTIONS	517
11.1.7	POST_READ_ACTIONS	518
11.1.8	POST_WRITE_ACTIONS	518
11.1.9	PRE_EDIT_ACTIONS	519
11.1.10	PRE_READ_ACTIONS	519
11.1.11	PRE_WRITE_ACTIONS	520
11.1.12	STYLE	520
11.1.13	Schémas de séquence pour les actions	522
11.2	EDIT_DISPLAY	524
11.2.1	Structure générale	524
11.2.2	EDIT_ITEMS	524
11.2.3	DISPLAY_ITEMS	525
11.2.4	POST_EDIT_ACTIONS	525
11.2.5	PRE_EDIT_ACTIONS	526
11.3	IMAGE	526
11.3.1	Structure générale	526
11.3.2	LINK	528
11.3.3	PATH	528
11.4	GRID	529
11.4.1	Structure générale	529
11.4.2	ORIENTATION	532
11.4.3	VECTORS	532
11.5	CHART	533
11.5.1	Structure générale	533
11.5.2	CYCLE_TIME	540
11.5.3	LENGTH	541
11.5.4	TYPE	541
11.5.5	Schéma de séquence pour les actions	544
11.5.6	Règles relatives aux légendes/aides de l'attribut CHART	544
11.6	SOURCE	547
11.6.1	Structure générale	547
11.6.2	EXIT_ACTIONS	548
11.6.3	INIT_ACTIONS	549
11.6.4	REFRESH_ACTIONS	549
11.6.5	Y_AXIS	550
11.7	AXIS	550
11.7.1	Structure générale	550
11.7.2	MIN_VALUE, MAX_VALUE	551
11.7.3	SCALING	552
11.7.4	CONSTANT_UNIT	552
11.8	GRAPH	553
11.8.1	Structure générale	553
11.8.2	CYCLE_TIME	561
11.8.3	X_AXIS	562
11.8.4	Schéma de séquence pour les actions	562
11.9	WAVEFORM	563
11.9.1	Structure générale	563

11.9.2	EXIT_ACTIONS	565
11.9.3	INIT_ACTIONS	565
11.9.4	KEY_POINTS	566
11.9.5	REFRESH_ACTIONS	567
11.9.6	TYPE	567
11.9.7	Y_AXIS	570
11.10	PLUGIN	571
11.10.1	Structure générale	571
11.10.2	UUID	572
12	METHOD	572
12.1	Structure générale	572
12.2	ACCESS	573
12.3	CLASS	574
12.4	TYPE	575
13	Modélisation d'équipement	576
13.1	BLOCK_A	576
13.1.1	Structure générale	576
13.1.2	AXIS_ITEMS	578
13.1.3	CHARACTERISTICS	579
13.1.4	CHARTS	579
13.1.5	CHART_ITEMS	580
13.1.6	COLLECTION_ITEMS	580
13.1.7	EDIT_DISPLAY_ITEMS	580
13.1.8	FILES	581
13.1.9	FILE_ITEMS	581
13.1.10	GRAPHS	581
13.1.11	GRAPH_ITEMS	582
13.1.12	GRIDS	582
13.1.13	GRID_ITEMS	582
13.1.14	IMAGE_ITEMS	583
13.1.15	ITEM_ARRAY_ITEMS	583
13.1.16	LISTS	583
13.1.17	LIST_ITEMS	584
13.1.18	LOCAL_PARAMETERS	584
13.1.19	MENUS	584
13.1.20	MENU_ITEMS	585
13.1.21	METHODS	585
13.1.22	METHOD_ITEMS	586
13.1.23	PARAMETERS	586
13.1.24	PARAMETER_LISTS	587
13.1.25	PLUGINS	587
13.1.26	PLUGIN_ITEMS	587
13.1.27	REFRESH_ITEMS	588
13.1.28	SEMANTIC_MAP_ITEMS	588
13.1.29	SOURCE_ITEMS	588
13.1.30	UNIT_ITEMS	589
13.1.31	WAVEFORM_ITEMS	589
13.1.32	WRITE_AS_ONE_ITEMS	589
13.2	BLOCK_B	590

13.2.1	Structure générale	590
13.2.2	NUMBER	590
13.2.3	TYPE	591
13.3	LIKE	591
13.4	IMPORT	592
13.5	TEMPLATE	596
13.5.1	Structure générale	596
13.5.2	DEFAULT_VALUES	597
13.6	COMPONENT	597
13.6.1	Structure générale	597
13.6.2	BYTE_ORDER	599
13.6.3	CAN_DELETE	600
13.6.4	CHECK_CONFIGURATION	600
13.6.5	COMPONENT_RELATIONS	600
13.6.6	CONNECTION_POINT	601
13.6.7	DETECT	601
13.6.8	EDD	601
13.6.9	INITIAL_VALUES	602
13.6.10	REDUNDANCY	602
13.6.11	SCAN	603
13.6.12	SCAN_LIST	603
13.6.13	PRODUCT_URI	603
13.7	COMPONENT_FOLDER	604
13.8	COMPONENT_REFERENCE	604
13.9	COMPONENT_RELATION	605
13.9.1	Structure générale	605
13.9.2	COMPONENTS	606
13.9.3	RELATION_TYPE	607
13.9.4	ADDRESSING	608
13.9.5	MAXIMUM_NUMBER	609
13.9.6	MINIMUM_NUMBER	609
13.9.7	REQUIRED_INTERFACE	609
13.10	INTERFACE	610
13.10.1	Structure générale	610
13.10.2	DECLARATION	610
14	Attributs communs	611
14.1	CLASSIFICATION	611
14.2	COMPONENT_PARENT	613
14.3	COMPONENT_PATH	613
14.4	DEFINITION	613
14.5	EMPHASIS	614
14.6	HANDLING	615
14.7	HEIGHT	615
14.8	HELP	616
14.9	LABEL	617
14.10	LINE_COLOR	617
14.11	LINE_TYPE	618
14.12	MEMBERS	619
14.13	PRIVATE	620

14.14	PROTOCOL	621
14.15	RESPONSE_CODES	621
14.16	SUPPLIED_INTERFACE	622
14.17	VALIDITY	622
14.18	VISIBILITY	623
14.19	WIDTH	623
14.20	WRITE_MODE	623
15	Redéfinitions	624
15.1	Vue d'ensemble	624
15.2	AXIS	627
15.3	BLOB	627
15.4	BLOCK_A	628
15.5	BLOCK_B	630
15.6	CHART	630
15.7	COLLECTION	631
15.8	COMMAND	631
15.9	COMPONENT	632
15.10	COMPONENT_FOLDER	632
15.11	COMPONENT_REFERENCE	633
15.12	COMPONENT_RELATION	633
15.13	EDIT_DISPLAY	634
15.14	FILE	634
15.15	GRAPH	635
15.16	GRID	635
15.17	IMAGE	636
15.18	INTERFACE	636
15.19	LIST	637
15.20	MENU	637
15.21	METHOD	638
15.22	PLUGIN	639
15.23	RECORD	639
15.24	REFERENCE_ARRAY	640
15.25	RESPONSE_CODES	640
15.26	SOURCE	641
15.27	TEMPLATE	641
15.28	VALUE_ARRAY	642
15.29	VARIABLE	642
15.30	VARIABLE_LIST	643
15.31	WAVEFORM	644
16	Référencement	645
16.1	Référencement d'une instance d'EDD	645
16.2	Référencement d'attribut AXIS de GRAPH, SOURCE, WAVEFORM	645
16.3	Référencement de bits de VARIABLE BIT_ENUMERATED	645
16.4	Référencement de membres de CHART	646
16.5	Référencement de membres de COLLECTION	646
16.6	Référencement d'instances de COMPONENT	646
16.7	Référencement de types de COMPONENT	647
16.8	Référencement de membres de FILE	647
16.9	Référencement de membres de GRAPH	648

16.10	Référencement d'éléments de LIST	648
16.11	Référencement de membres de RECORD	648
16.12	Référencement d'éléments de REFERENCE_ARRAY	649
16.13	Référencement de membres de SOURCE	649
16.14	Référencement d'éléments de VALUE_ARRAY	649
16.15	Référencement de membres de VARIABLE_LISTS	650
16.16	Référencement de CHARACTERISTICS de BLOCK_A	650
16.17	Référencement d'éléments de LOCAL_PARAMETERS de BLOCK_A	650
16.18	Référencement d'éléments de PARAMETERS de BLOCK_A	651
16.19	Référencement d'éléments de PARAMETER_LISTS de BLOCK_A	651
16.20	Référencement de CHARACTERISTICS d'une instance spécifique de BLOCK_A	651
16.21	Référencement de CHARTS d'une instance spécifique de BLOCK_A	652
16.22	Référencement de FILES d'une instance spécifique de BLOCK_A	652
16.23	Référencement de GRAPHS d'une instance spécifique de BLOCK_A	653
16.24	Référencement de GRIDS d'une instance spécifique de BLOCK_A	653
16.25	Référencement de LISTS d'une instance spécifique de BLOCK_A	654
16.26	Référencement de LOCAL_PARAMETERS d'une instance spécifique de BLOCK_A	654
16.27	Référencement de MENUS d'une instance spécifique de BLOCK_A	655
16.28	Référencement de METHODS d'une instance spécifique de BLOCK_A	655
16.29	Référencement de PARAMETERS d'une instance spécifique de BLOCK_A	656
16.30	Référencement de PLUGINS d'une instance spécifique de BLOCK_A	656
17	Chaînes	657
17.1	Spécification d'une chaîne sous la forme d'un littéral de chaîne	657
17.2	Spécification d'une chaîne en tant que variable de chaîne	657
17.3	Spécification d'une chaîne en tant que valeur d'énumération	657
17.4	Spécification d'une chaîne en tant que référence de dictionnaire	658
17.5	Référencement d'attributs HELP et LABEL d'instances d'EDD	658
17.6	Opérations de chaîne	659
18	Expression conditionnelle	659
18.1	Structure générale	659
18.2	Instruction conditionnelle IF	660
18.3	Instruction conditionnelle SELECT	660
19	Expression	661
19.1	Structure générale	661
19.2	Expressions primaires	661
19.3	Expressions unaires	664
19.4	Expressions binaires	665
19.4.1	Structure générale	665
19.4.2	Opérateurs multiplicatifs	665
19.4.3	Opérateurs additifs	665
19.4.4	Opérateurs de décalage	666
19.4.5	Opérateurs relationnels	666
19.4.6	Opérateurs d'égalité	666
19.4.7	Opérateur ET binaire (&)	667
19.4.8	Opérateur XOU binaire (^)	667
19.4.9	Opérateur OU binaire ()	667
19.4.10	Opérateur ET logique (&&)	667

19.4.11	Opérateur OU logique ()	667
19.4.12	Evaluation conditionnelle	667
20	Dictionnaire de texte	667
Annexe A (normative) Définition formelle d'EDDL		669
A.1	Préprocesseur EDDL	669
A.1.1	Structure générale	669
A.1.2	Directives	669
A.1.3	Macros prédéfinies	672
A.1.4	Caractères NEWLINE	673
A.1.5	Commentaires	673
A.2	Conventions	673
A.2.1	Constantes entières	673
A.2.2	Constantes à virgule flottante	673
A.2.3	Littéraux de chaîne	674
A.2.4	Utilisation de codes de langue et de pays dans des littéraux de chaîne	675
A.3	Opérateurs	676
A.4	Mots-clés	679
A.5	Symboles et leur domaine d'application	683
A.6	Terminaux	684
A.7	Syntaxe EDDL formelle	684
A.7.1	Généralités	684
A.7.2	Informations d'identification d'une EDD	685
A.7.3	AXIS	686
A.7.4	BLOCK_A et BLOCK_B	687
A.7.5	CHART	692
A.7.6	COLLECTION	692
A.7.7	COMMAND	693
A.7.8	COMPONENT	696
A.7.9	COMPONENT_FOLDER	699
A.7.10	COMPONENT_REFERENCE	700
A.7.11	COMPONENT_RELATION	700
A.7.12	EDIT_DISPLAY	702
A.7.13	FILE	703
A.7.14	GRAPH	703
A.7.15	GRID	704
A.7.16	IMAGE	705
A.7.17	INTERFACE	706
A.7.18	LIST	706
A.7.19	IMPORT	706
A.7.20	LIKE	709
A.7.21	MENU	710
A.7.22	METHOD	712
A.7.23	RECORD	714
A.7.24	REFERENCE_ARRAY	714
A.7.25	Relations	715
A.7.26	RESPONSE_CODES	717
A.7.27	SOURCE	717
A.7.28	TEMPLATE	718
A.7.29	VALUE_ARRAY	718

A.7.30	VARIABLE	719
A.7.31	VARIABLE_LIST	729
A.7.32	WAVEFORM	729
A.7.33	Attributs communs	731
A.7.34	Expression	735
A.7.35	Syntaxe C	738
A.7.36	Redéfinition	742
A.7.37	Références	766
A.7.38	PLUGIN	769
A.7.39	BLOB	769
A.7.40	SEMANTIC_MAP	770
A.8	Syntaxe de dictionnaire formelle	770
Annexe B (normative)	Types de données	772
B.1	Types de données de DEFINITION de METHOD	772
B.2	Types de données de TYPE de VARIABLE	773
B.2.1	Généralités	773
B.2.2	Codage des données DATE	775
B.2.3	Codage des données DATE_AND_TIME	775
B.2.4	Codage des données DURATION	776
B.2.5	Codage des données TIME	777
B.2.6	Codage des données TIME_VALUE	777
B.2.7	Codage du FORMAT DE DONNEES PACKED_ASCII (ASCII 6 BITS)	778
B.2.8	Codage de l'attribut BOOLEAN	779
B.2.9	Codage de l'attribut BIT_ENUMERATED	779
Annexe C (informative)	Exemple d'EDD	781
C.1	Exemple d'EDD d'un transmetteur de température	781
C.2	Exemple d'EDD	782
Bibliographie		795
Figure 1	– Position de l'IEC 61804 par rapport à d'autres normes et produits	416
Figure 2	– Processus de génération d'EDD	423
Figure 3	– BLOCK_A	428
Figure 4	– CHART	429
Figure 5	– COLLECTION	429
Figure 6	– COMMAND	430
Figure 7	– COMPONENT	430
Figure 8	– COMPONENT_FOLDER	431
Figure 9	– COMPONENT_REFERENCE	431
Figure 10	– COMPONENT_RELATION	431
Figure 11	– EDIT_DISPLAY	431
Figure 12	– FILE	432
Figure 13	– GRAPH	432
Figure 14	– GRID	432
Figure 15	– IMAGE	432
Figure 16	– LIKE	433
Figure 17	– LIST	434

Figure 18 – MENU	434
Figure 19 – RECORD	435
Figure 20 – REFERENCE_ARRAY	435
Figure 21 – REFRESH	435
Figure 22 – SEMANTIC_MAP	436
Figure 23 – UNIT	436
Figure 24 – WRITE_AS_ONE	436
Figure 25 – SOURCE	437
Figure 26 – VALUE_ARRAY	437
Figure 27 – VARIABLE	438
Figure 28 – VARIABLE_LIST	438
Figure 29 – WAVEFORM	438
Figure 30 – Exemple d'attribut REFERENCE_ARRAY	478
Figure 31 – Exemple de définition de commande abstraite	478
Figure 32 – Exemples d'EDD d'attributs SEMANTIC_MAP	493
Figure 33 – Exemple de masque d'élément	507
Figure 34 – Exemple de masque d'élément simple	507
Figure 35 – Exemple de mapping avec un masque d'élément simple	507
Figure 36 – Exemple de masque d'élément multiple	508
Figure 37 – Exemple de mapping avec un masque d'élément multiple	508
Figure 38 – Qualificatif INFO	509
Figure 39 – Qualificatif INDEX	509
Figure 40 – Qualificatifs INFO et INDEX	509
Figure 41 – Exemples d'attributs MENU_STYLE	521
Figure 42 – Activation de MENU	523
Figure 43 – Exemple d'EDD d'un attribut IMAGE	527
Figure 44 – Exemple d'EDD d'un attribut IMAGE avec l'attribut LINK	528
Figure 45 – Exemple d'EDD d'un attribut GRID	531
Figure 46 – Résultats de l'exemple d'EDD	531
Figure 47 – Exemple de diagramme avec une courbe dans une boîte de dialogue	535
Figure 48 – Exemple de diagramme avec deux attributs SOURCE	536
Figure 49 – Affichage d'un exemple de diagramme avec deux attributs SOURCE	537
Figure 50 – Exemple de diagramme avec trois barres horizontales	538
Figure 51 – Affichage d'un exemple de diagramme avec trois barres horizontales	538
Figure 52 – Exemple de diagramme dans une boîte de dialogue	540
Figure 53 – Exemple de types d'attributs CHART de tendances	543
Figure 54 – Exemple de types d'attributs CHART d'instruments de mesure	543
Figure 55 – Activation de l'attribut CHART	544
Figure 56 – Légende/aide pour les attributs SOURCE avec des attributs VARIABLE simples et multiples – Attribut Y_AXIS non défini	545
Figure 57 – Légende/aide pour les attributs SOURCE – Disposition de plusieurs attributs SOURCE avec un attribut Y_AXIS partagé	547
Figure 58 – Graphique et éléments visuels	554
Figure 59 – Exemple de graphique	557

Figure 60 – Axes multiusages	558
Figure 61 – EDD avec zoom et défilement pris en charge par l'équipement.....	561
Figure 62 – Activation de l'attribut GRAPH.....	563
Figure 63 – Mécanismes d'importation EDDL.....	593
Figure 64 – Exemples d'utilisation de l'attribut TEMPLATE	597
Figure 65 – Utilisation de l'attribut REQUIRED_RANGES	607
Figure 66 – Utilisation de l'attribut NEXT_COMPONENT.....	608
Figure C.1 – Exemple d'écran d'opérateur pour l'utilisation d'une EDD	781
Tableau 1 – Format du tableau des constructions de base.....	424
Tableau 2 – Contenu du tableau des constructions de base.....	424
Tableau 3 – Description de l'utilisation des attributs.....	425
Tableau 4 – Règles de convention lexicale	426
Tableau 5 – Sources EDD.....	426
Tableau 6 – Attributs d'informations d'identification d'EDD.....	441
Tableau 7 – Attribut DD_REVISION	441
Tableau 8 – Attribut DEVICE_REVISION	442
Tableau 9 – Attribut DEVICE_TYPE.....	442
Tableau 10 – Attribut EDD_PROFILE.....	443
Tableau 11 – Attribut EDD_VERSION	443
Tableau 12 – Attribut LAYOUT_TYPE.....	444
Tableau 13 – Attribut MANUFACTURER.....	444
Tableau 14 – Attribut MANUFACTURER_EXT	445
Tableau 15 – Attributs VARIABLE	446
Tableau 16 – Attributs CLASS	447
Tableau 17 – Attribut CONSTANT_UNIT.....	450
Tableau 18 – Attributs POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS.....	452
Tableau 19 – Attributs POST_USERCHANGE_ACTIONS et POST_RQSTUPDATE_ACTIONS	453
Tableau 20 – Présentation des attributs TYPE	456
Tableau 21 – Attributs MIN_VALUE et MAX_VALUE	457
Tableau 22 – Attributs TYPE pour DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER.....	459
Tableau 23 – Type de conversion du format d'affichage par défaut.....	460
Tableau 24 – Largeur/précision par défaut fondées sur l'attribut VARIABLE et le type de conversion de format	460
Tableau 25 – Attributs TYPE pour DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE.....	464
Tableau 26 – Attributs TYPE pour BIT_ENUMERATED.....	465
Tableau 27 – Attribut size	466
Tableau 28 – Attributs status-class	467
Tableau 29 – Attributs ALL, AO, DV, TV	468
Tableau 30 – Attributs TYPE pour ENUMERATED	470
Tableau 31 – Attributs TYPE pour INDEX	471

Tableau 32 – Attributs TYPE pour ASCII, EUC, PACKED_ASCII, PASSWORD, VISIBLE	473
Tableau 33 – Gestion de chaîne	473
Tableau 34 – Attributs TYPE pour OCTET	474
Tableau 35 – Attributs TYPE pour BITSTRING.....	476
Tableau 36 – Attributs REFERENCE_ARRAY	477
Tableau 37 – Attribut ELEMENTS	479
Tableau 38 – Attributs VALUE_ARRAY	480
Tableau 39 – Attribut NUMBER_OF_ELEMENTS	481
Tableau 40 – Attribut TYPE	481
Tableau 41 – Attributs LIST	482
Tableau 42 – Attribut CAPACITY	483
Tableau 43 – Attribut COUNT	484
Tableau 44 – Attribut TYPE	485
Tableau 45 – Attributs COLLECTION	486
Tableau 46 – item-type	486
Tableau 47 – Attributs FILE	488
Tableau 48 – Attribut IDENTITY.....	489
Tableau 49 – Attribut ON_UPDATE_ACTIONS.....	489
Tableau 50 – Attribut SHARED	490
Tableau 51 – Présentation de la relation REFRESH.....	490
Tableau 52 – Attributs REFRESH	491
Tableau 53 – Présentation de la relation SEMANTIC_MAP	492
Tableau 54 – Attributs SEMANTIC_MAP.....	492
Tableau 55 – Présentation de la relation UNIT	494
Tableau 56 – Attributs UNIT.....	494
Tableau 57 – Présentation de la relation WRITE_AS_ONE	495
Tableau 58 – Attribut WRITE_AS_ONE.....	495
Tableau 59 – Attributs BLOB.....	496
Tableau 60 – Attribut IDENTITY.....	496
Tableau 61 – Attributs COMMAND.....	498
Tableau 62 – Attribut API.....	499
Tableau 63 – Attribut BLOCK_B.....	500
Tableau 64 – Attribut HEADER	500
Tableau 65 – Attribut INDEX	500
Tableau 66 – Attribut NUMBER.....	501
Tableau 67 – Attribut OPERATION	502
Tableau 68 – Mapping des communications PROFIBUS et PROFINET	502
Tableau 69 – Attribut SLOT	502
Tableau 70 – Attribut SUB_SLOT.....	503
Tableau 71 – Attributs TRANSACTION	504
Tableau 72 – Attributs REPLY et REQUEST	505
Tableau 73 – Attributs POST_RQSTRECEIVE_ACTIONS	510

Tableau 74 – Attributs RECORD	511
Tableau 75 – Présentation de l'attribut RESPONSE_CODES	511
Tableau 76 – Attributs RESPONSE_CODES	512
Tableau 77 – Attributs VARIABLE_LIST	513
Tableau 78 – Attributs MENU	514
Tableau 79 – Attribut ACCESS	514
Tableau 80 – Attributs MENU ITEMS	516
Tableau 81 – Qualificatifs MENU ITEMS	517
Tableau 82 – Attributs EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS	518
Tableau 83 – Attribut STYLE	521
Tableau 84 – Attributs EDIT_DISPLAY	524
Tableau 85 – Attribut EDIT_ITEMS	525
Tableau 86 – Attributs DISPLAY_ITEM	525
Tableau 87 – Attributs POST_EDIT_ACTIONS et PRE_EDIT_ACTIONS	526
Tableau 88 – Attributs IMAGE	527
Tableau 89 – Attribut LINK	528
Tableau 90 – Attribut PATH	529
Tableau 91 – Formats d'image	529
Tableau 92 – Attributs GRID	530
Tableau 93 – Attribut ORIENTATION	532
Tableau 94 – Attribut VECTORS	533
Tableau 95 – Attributs CHART	534
Tableau 96 – Attribut CYCLE_TIME	541
Tableau 97 – Attribut LENGTH	541
Tableau 98 – Attribut TYPE	542
Tableau 99 – Attributs SOURCE	548
Tableau 100 – Attribut Y_AXIS	550
Tableau 101 – Attributs AXIS	551
Tableau 102 – Attributs MIN_VALUE et MAX_VALUE	552
Tableau 103 – Attribut SCALING	552
Tableau 104 – Attributs GRAPH	555
Tableau 105 – Attribut CYCLE_TIME	562
Tableau 106 – Attribut X_AXIS	562
Tableau 107 – Attributs WAVEFORM	564
Tableau 108 – Attributs EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS	565
Tableau 109 – Attribut KEY_POINTS	566
Tableau 110 – Attributs X_VALUES et Y_VALUES	567
Tableau 111 – Attribut TYPE	568
Tableau 112 – Attribut XY	568
Tableau 113 – Attributs YT	569
Tableau 114 – Attribut HORIZONTAL	570
Tableau 115 – Attribut VERTICAL	570

Tableau 116 – Attribut Y_AXIS	571
Tableau 117 – Attributs PLUGIN	571
Tableau 118 – Attribut UUID	572
Tableau 119 – Attributs METHOD	573
Tableau 120 – Types de paramètres	573
Tableau 121 – Attributs ACCESS	574
Tableau 122 – Attributs CLASS	574
Tableau 123 – Attributs TYPE	576
Tableau 124 – Attributs BLOCK_A	578
Tableau 125 – Attribut AXIS_ITEMS	579
Tableau 126 – Attribut CHARACTERISTICS	579
Tableau 127 – Attribut CHARTS	579
Tableau 128 – Attribut CHART_ITEMS	580
Tableau 129 – Attribut COLLECTION_ITEMS	580
Tableau 130 – Attribut EDIT_DISPLAY_ITEMS	580
Tableau 131 – Attribut FILES	581
Tableau 132 – Attribut FILE_ITEMS	581
Tableau 133 – Attribut GRAPHS	582
Tableau 134 – Attribut GRAPH_ITEMS	582
Tableau 135 – Attribut GRIDS	582
Tableau 136 – Attribut GRID_ITEMS	583
Tableau 137 – Attribut IMAGE_ITEMS	583
Tableau 138 – Attribut ITEM_ARRAY_ITEMS	583
Tableau 139 – Attribut LISTS	584
Tableau 140 – Attribut LIST_ITEMS	584
Tableau 141 – Attribut MENUS	585
Tableau 142 – Attribut MENU_ITEMS	585
Tableau 143 – Attribut METHODS	585
Tableau 144 – Attribut METHOD_ITEMS	586
Tableau 145 – Attributs PARAMETER	586
Tableau 146 – Attributs PARAMETER_LISTS	587
Tableau 147 – Attribut PLUGINS	587
Tableau 148 – Attribut PLUGIN_ITEMS	588
Tableau 149 – Attribut REFRESH_ITEMS	588
Tableau 150 – Attribut SEMANTIC_MAP_ITEMS	588
Tableau 151 – Attribut SOURCE_ITEMS	589
Tableau 152 – Attribut UNIT_ITEMS	589
Tableau 153 – Attribut WAVEFORM_ITEMS	589
Tableau 154 – Attribut WRITE_AS_ONE_ITEMS	590
Tableau 155 – Attributs BLOCK_B	590
Tableau 156 – Attributs NUMBER	591
Tableau 157 – Attributs TYPE	591
Tableau 158 – Présentation de l'attribut LIKE	592

Tableau 159 – Attributs LIKE	592
Tableau 160 – Présentation de l'attribut IMPORT	594
Tableau 161 – Importation de description d'un équipement	594
Tableau 162 – Attributs TEMPLATE	596
Tableau 163 – Attribut DEFAULT_VALUES	597
Tableau 164 – Attributs COMPONENT	599
Tableau 165 – Attribut BYTE_ORDER	599
Tableau 166 – Attribut CAN_DELETE	600
Tableau 167 – Attribut CHECK_CONFIGURATION	600
Tableau 168 – Attribut COMPONENT_RELATIONS	601
Tableau 169 – Attribut CONNECTION_POINT	601
Tableau 170 – Attribut DETECT	601
Tableau 171 – Attribut EDD	602
Tableau 172 – Attribut INITIAL_VALUES	602
Tableau 173 – Attribut REDUNDANCY	603
Tableau 174 – Attribut SCAN	603
Tableau 175 – Attribut SCAN_LIST	603
Tableau 176 – Attribut PRODUCT_URI	604
Tableau 177 – Attributs COMPONENT_FOLDER	604
Tableau 178 – Attributs COMPONENT_REFERENCE	605
Tableau 179 – Attributs COMPONENT_RELATION	606
Tableau 180 – Attribut COMPONENTS	607
Tableau 181 – Attribut RELATION_TYPE	608
Tableau 182 – Attribut ADDRESSING	608
Tableau 183 – Attribut MAXIMUM_NUMBER	609
Tableau 184 – Attribut MINIMUM_NUMBER	609
Tableau 185 – Attribut REQUIRED_INTERFACE	609
Tableau 186 – Attributs INTERFACE	610
Tableau 187 – Attribut DECLARATION	611
Tableau 188 – Attributs CLASSIFICATION	611
Tableau 189 – Attribut COMPONENT_PARENT	613
Tableau 190 – Attribut COMPONENT_PAT	613
Tableau 191 – Attribut DEFINITION	614
Tableau 192 – Attribut EMPHASIS	615
Tableau 193 – Attribut HANDLING	615
Tableau 194 – Attributs HEIGHT/WIDTH	616
Tableau 195 – Attribut HELP	617
Tableau 196 – Attribut LABEL	617
Tableau 197 – Attribut LINE_COLOR	618
Tableau 198 – Valeurs des couleurs RVB normalisées	618
Tableau 199 – Attribut LINE_TYPE	619
Tableau 200 – Attributs MEMBERS	620
Tableau 201 – Attributs PRIVATE	620

Tableau 202 – Attribut PROTOCOL	621
Tableau 203 – Attribut RESPONSE_CODES.....	621
Tableau 204 – Attribut SUPPLIED_INTERFACE	622
Tableau 205 – Attributs VALIDITY	623
Tableau 206 – Attributs VISIBILITY.....	623
Tableau 207 – Attributs WRITE_MODE.....	624
Tableau 208 – Attributs de redéfinition.....	627
Tableau 209 – Règles de redéfinition pour les attributs AXIS	627
Tableau 210 – Règles de redéfinition pour les attributs BLOB.....	628
Tableau 211 – Règles de redéfinition pour les attributs BLOCK_A	629
Tableau 212 – Règles de redéfinition pour les attributs BLOCK_B	630
Tableau 213 – Règles de redéfinition pour les attributs CHART	630
Tableau 214 – Règles de redéfinition pour les attributs COLLECTION	631
Tableau 215 – Règles de redéfinition pour les attributs COMMAND	631
Tableau 216 – Règles de redéfinition pour les attributs COMPONENT	632
Tableau 217 – Règles de redéfinition pour les attributs COMPONENT_FOLDER	633
Tableau 218 – Règles de redéfinition pour les attributs COMPONENT_REFERENCE	633
Tableau 219 – Règles de redéfinition pour les attributs COMPONENT_RELATION	634
Tableau 220 – Règles de redéfinition pour les attributs EDIT_DISPLAY.....	634
Tableau 221 – Règles de redéfinition pour les attributs FILE	635
Tableau 222 – Règles de redéfinition pour les attributs GRAPH.....	635
Tableau 223 – Règles de redéfinition pour les attributs GRID	636
Tableau 224 – Règles de redéfinition pour les attributs IMAGE.....	636
Tableau 225 – Règles de redéfinition pour les attributs INTERFACE	637
Tableau 226 – Règles de redéfinition pour les attributs LIST	637
Tableau 227 – Règles de redéfinition pour les attributs MENU	638
Tableau 228 – Règles de redéfinition pour les attributs METHOD	638
Tableau 229 – Règles de redéfinition pour les attributs PLUGIN	639
Tableau 230 – Règles de redéfinition pour les attributs RECORD	639
Tableau 231 – Règles de redéfinition pour les attributs REFERENCE_ARRAY	640
Tableau 232 – Règles de redéfinition pour les attributs RESPONSE_CODES	640
Tableau 233 – Règles de redéfinition pour les attributs SOURCE	641
Tableau 234 – Règles de redéfinition pour les attributs TEMPLATE	641
Tableau 235 – Règles de redéfinition pour les attributs VALUE_ARRAY	642
Tableau 236 – Règles de redéfinition pour les attributs VARIABLE	643
Tableau 237 – Règles de redéfinition pour les attributs VARIABLE_LIST	644
Tableau 238 – Règles de redéfinition pour les attributs WAVEFORM	644
Tableau 239 – Référencement d'une instance d'EDD	645
Tableau 240 – Référencement de l'attribut AXIS d'un attribut GRAPH, SOURCE, WAVEFORM	645
Tableau 241 – Référencement d'éléments de VARIABLE	646
Tableau 242 – Référencement de membres de CHART	646
Tableau 243 – Référencement de membres de COLLECTION	646

Tableau 244 – Référencement d'une instance de COMPONENT	647
Tableau 245 – Référencement d'un type de COMPONENT	647
Tableau 246 – Référencement de membres de FILE	647
Tableau 247 – Référencement de membres de GRAPH	648
Tableau 248 – Référencement d'éléments de LIST	648
Tableau 249 – Référencement d'éléments de RECORD	649
Tableau 250 – Référencement de membres de REFERENCE_ARRAY	649
Tableau 251 – Référencement de membres de SOURCE	649
Tableau 252 – Référencement d'éléments de VALUE_ARRAY	650
Tableau 253 – Référencement de membres de VARIABLE_LISTS	650
Tableau 254 – Référencement de CHARACTERISTICS de BLOCK_A	650
Tableau 255 – Référencement de membres de LOCAL_PARAMETERS de BLOCK_A	651
Tableau 256 – Référencement de membres de PARAMETERS de BLOCK_A	651
Tableau 257 – Référencement de membres de PARAMETER_LISTS de BLOCK_A	651
Tableau 258 – Référencement de CHARACTERISTICS d'une instance spécifique de BLOCK_A	652
Tableau 259 – Référencement de CHARTS d'une instance spécifique de BLOCK_A	652
Tableau 260 – Référencement de FILES d'une instance spécifique de BLOCK_A	653
Tableau 261 – Référencement de GRAPHS d'une instance spécifique de BLOCK_A	653
Tableau 262 – Référencement de GRIDS d'une instance spécifique de BLOCK_A	654
Tableau 263 – Référencement de LISTS d'une instance spécifique de BLOCK_A	654
Tableau 264 – Référencement de LOCAL_PARAMETERS d'une instance spécifique de BLOCK_A	655
Tableau 265 – Référencement de MENUS d'une instance spécifique de BLOCK_A	655
Tableau 266 – Référencement de METHODS d'une instance spécifique de BLOCK_A	656
Tableau 267 – Référencement de PARAMETERS d'une instance spécifique de BLOCK_A	656
Tableau 268 – Référencement de PLUGINS d'une instance spécifique de BLOCK_A	657
Tableau 269 – Chaîne sous forme de littéral de chaîne	657
Tableau 270 – Chaîne en tant que variable de chaîne	657
Tableau 271 – Chaîne en tant que valeur d'énumération	658
Tableau 272 – Chaîne en tant que référence de dictionnaire	658
Tableau 273 – Référencement des attributs HELP et LABEL des instances d'EDD	659
Tableau 274 – Opération de chaîne	659
Tableau 275 – Instruction conditionnelle IF	660
Tableau 276 – Instruction conditionnelle SELECT	661
Tableau 277 – Expressions primaires	662
Tableau 278 – Valeurs d'attribut VARIABLE	663
Tableau 279 – Valeurs d'attribut AXIS	664
Tableau 280 – Valeurs d'attribut BLOB	664
Tableau 281 – Valeurs d'attribut LIST	664
Tableau 282 – Valeurs d'attribut ARRAY	664
Tableau 283 – Expressions unaires	665
Tableau 284 – Opérateurs multiplicatifs	665

Tableau 285 – Opérateurs additifs	666
Tableau 286 – Opérateurs de décalage	666
Tableau 287 – Opérateurs relationnels	666
Tableau 288 – Opérateurs d'égalité	666
Tableau 289 – Attributs de dictionnaire de texte	668
Tableau A.1 – Conventions pour les constantes entières	673
Tableau A.2 – Utilisation de séquences d'échappement dans des littéraux de chaîne	674
Tableau A.3 – Exemples de codes de langue pour les littéraux de chaîne.....	675
Tableau A.4 – Priorité et associativité pour les opérateurs EDDL.....	676
Tableau A.5 – Opérations pour les attributs VARIABLE ou les variables locales METHOD	678
Tableau A.6 – Mots-clés EDDL	679
Tableau B.1 – Types de données de DEFINITION de METHOD	772
Tableau B.2 – Attributs TYPE de VARIABLE	773
Tableau B.3 – Codage de l'attribut DATE	775
Tableau B.4 – Codage de l'attribut DATE_AND_TIME.....	776
Tableau B.5 – Codage de l'attribut DURATION	776
Tableau B.6 – Codage de l'attribut DURATION pour ISA100.....	776
Tableau B.7 – Codage de l'attribut TIME.....	777
Tableau B.8 – Codage de l'attribut TIME pour ISA100.....	777
Tableau B.9 – Codage de l'attribut TIME_VALUE (quatre octets)	777
Tableau B.10 – Codage de l'attribut TIME_VALUE (huit octets)	778
Tableau B.11 – Codage des caractères PACKED_ASCII.....	779
Tableau B.12 – Codage de l'attribut BOOLEAN.....	779
Tableau B.13 – Codage de l'attribut BIT_ENUMERATED	780
Tableau B.14 – Codage de l'attribut BIT_ENUMERATED pour FF	780

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

**LES DISPOSITIFS ET LEUR INTÉGRATION DANS LES SYSTÈMES
DE L'ENTREPRISE – BLOCS FONCTIONNELS (FB) POUR
LES PROCÉDÉS 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 dispositifs 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 quatrième édition annule et remplace la troisième édition parue en 2015. Cette édition constitue une révision technique.

Cette édition a été développée en fusionnant le contenu de plusieurs variantes des spécifications de l'EDDL existantes, y compris celles du FieldComm Group (FOUNDATION™ Fieldbus¹, HART®²), du PROFIBUS™³ Nutzerorganisation e.V. (PNO), et de l'ISA100_Wireless™⁴ Compliance Institute (ISA100 WCI). Les endroits pouvant comporter une déviation de profil sont désormais indiqués dans le contexte où la déviation en question a été identifiée. Par conséquent, le formatage et la numérotation de la présente édition peuvent différer des spécifications individuelles desquelles elle est issue.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

- ajout des profils de communication ISA100 et GPE;
- ajout du nouvel attribut LAYOUT_TYPE aux informations d'identification de l'EDD;
- ajout de la nouvelle construction SEMANTIC_MAP;
- ajout des valeurs LOCAL_A et LOCAL_B de l'attribut CLASS;
- extension de la fonctionnalité LIST pour prendre en charge les listes d'équipements gérées.

Le texte de cette norme est issu des documents suivants:

CDV	Rapport de vote
65E/631/CDV	65E/689/RVC

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 *Les dispositifs et leur intégration dans les systèmes de l'entreprise – 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.

¹ FOUNDATION™ Fieldbus est l'appellation commerciale du FieldComm Group. Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

² HART® est une marque déposée du FieldComm Group. Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

³ PROFIBUS et PROFINET sont les appellations commerciales du PROFIBUS Nutzerorganisation e.V. Cette information est donnée à l'intention des utilisateurs du présent document et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

⁴ ISA100_Wireless™ est l'appellation commerciale de l'ISA100 Wireless Compliance Institute. Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

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.

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020

INTRODUCTION

Le langage EDDL établit un lien entre la spécification conceptuelle de bloc fonctionnel de l'IEC 61804-2 et une mise en œuvre de produit. Il permet aux fabricants d'utiliser la même méthode de description pour des équipements fondés sur différentes technologies et plateformes. La Figure 1 représente 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 builtin EDDL

Partie 6: Meeting the requirements for integrating fieldbus devices in engineering tools for field devices (disponible en anglais seulement)

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 équipements 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.

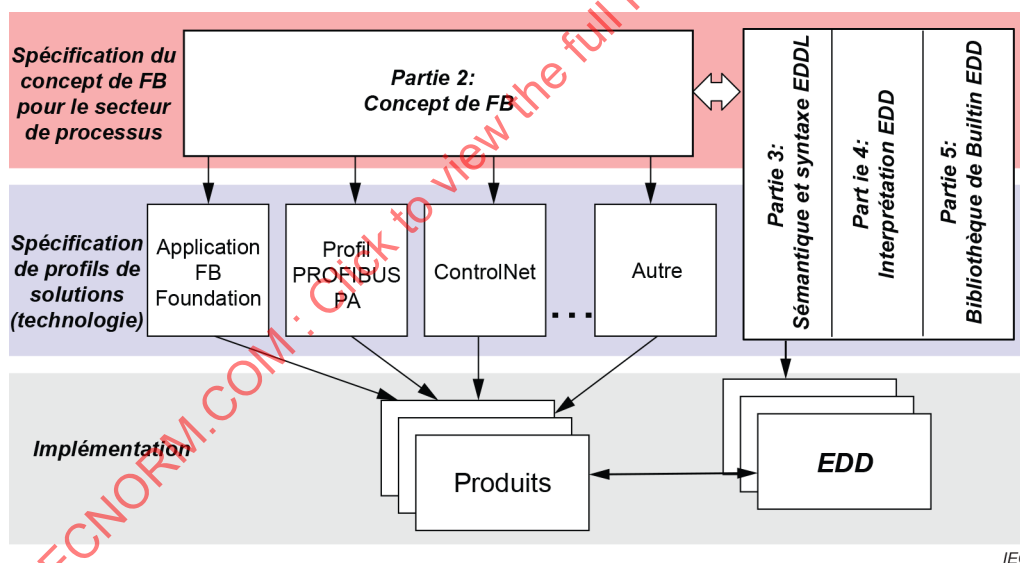


Figure 1 – Position de l'IEC 61804 par rapport à d'autres normes et produits

LES DISPOSITIFS ET LEUR INTÉGRATION DANS LES SYSTÈMES DE L'ENTREPRISE – BLOCS FONCTIONNELS (FB) POUR LES PROCÉDÉS INDUSTRIELS ET LE LANGAGE DE DESCRIPTION ÉLECTRONIQUE DE PRODUIT (EDDL) –

Partie 3: Sémantique et syntaxe EDDL

1 Domaine d'application

La présente partie de l'IEC 61804 spécifie la technologie du langage de description électronique de produit (EDDL), laquelle permet l'intégration d'informations de produits réels à l'aide des outils du cycle de vie d'ingénierie.

Le présent document spécifie l'EDDL en tant que langage générique pour décrire les propriétés des composants de systèmes d'automatisation. L'EDDL est capable de décrire

- les paramètres des équipements et leurs dépendances;
- les fonctions des équipements, par exemple le mode de simulation, l'étalonnage;
- les représentations graphiques, par exemple les menus;
- les interactions avec les équipements de commande;
- les représentations graphiques:
 - interface utilisateur avancée;
 - système graphique;
- le répertoire des données persistantes.

L'EDDL est utilisé pour créer une description électronique de produit (EDD) pour, par exemple, des équipements concrets, des profils communs ou des bibliothèques. Cette EDD est utilisée avec des outils appropriés pour générer un code interprétatif qui prend en charge la gestion des paramètres, l'exploitation et la surveillance de composants de système d'automatisation, tels que des E/S, des contrôleurs, des capteurs et des contrôleurs programmables à distance. La mise en œuvre des outils ne fait pas partie du domaine d'application du présent document.

Le présent document spécifie la sémantique et la structure lexicale indépendamment de la syntaxe. Une syntaxe spécifique est définie à l'Annexe A, mais il est possible d'utiliser également le modèle sémantique avec d'autres syntaxes.

L'IEC 61804-4 spécifie l'interprétation EDD pour les applications EDD et les EDD afin d'assurer l'interopérabilité EDD.

L'IEC 61804-5 spécifie la bibliothèque de builtin EDDL et fournit les profils des différents bus de terrain.

2 Références normatives

Les documents suivants cités dans le texte constituent, pour tout ou partie de leur contenu, des exigences du présent document. 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 60050-351, *Vocabulaire Electrotechnique International (IEV) – Partie 351: Technologie de commande et de régulation*

IEC 61804-2, *Function blocks (FB) for process control – Part 2: Specification of FB concept* (disponible en anglais seulement)

IEC 61804-4, *Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL) – Partie 4: Interprétation EDD*

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 9899, *Langages de programmation – C*

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, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1* (disponible en anglais seulement)

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* (disponible en anglais seulement)

ISO/IEC 9899:1999, *Programming languages – C* (disponible en anglais seulement)

ISO/IEC 10646, *Information technology – Universal Coded Character Set (UCS)* (disponible en anglais seulement)

ISO/IEC 10918-1, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines* (disponible en anglais seulement)

ISO/IEC 15948, *Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification* (disponible en anglais seulement)

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*

ISO/TS 29002-5, *Industrial automation systems and integration – Exchange of characteristic data – Part 5: Identification scheme* (disponible en anglais seulement)

IEEE 754, *IEEE Standard for Floating-Point Arithmetic* (disponible en anglais seulement)

RFC 3629, *User Datagram Protocol* (disponible en anglais seulement), disponible à l'adresse <http://www.ietf.org/rfc/rfc0768.txt> [consulté 2018-12-21]

W3C Recommendation, *Cascading Style Sheets Level 2 Specification* (disponible en anglais seulement), disponible à l'adresse <http://www.w3.org/TR/CSS2> [consulté 2018-12-21]

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 et l'IEC 61804-2, ainsi que les suivants s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- ISO Online browsing platform: disponible à l'adresse <https://www.iso.org/obp>
- IEC Electropedia: disponible à l'adresse <http://www.electropedia.org/>

3.1.1

builtin

sous-programme prédéfini exécuté par l'application EDD, pour la communication et l'affichage, par exemple

3.1.2

équipement

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 équipements 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 électronique de produit EDD

collection de données contenant les paramètres de l'équipement, leurs dépendances, leur représentation graphique et une description des ensembles de données qui sont transférés

Note 1 à l'article: La description électronique de produit 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 électronique de produit

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 électronique de produit 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

bloc fonctionnel

instance de bloc fonctionnel

unité fonctionnelle logicielle consistant en une copie individuelle nommée d'une structure de données et des opérations associées spécifiées par un type de bloc fonctionnel correspondant

Note 1 à l'article: Les opérations types d'un FB comprennent la modification des valeurs des données dans sa structure de données associée.

[SOURCE: IEC 61804-2:2018, 3.1.35]

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 d'écrire des macros pour rendre le code plus facile à lire.

3.1.14

logiciel

création intellectuelle comprenant les programmes, procédures, règles et toute documentation associée relatifs à l'exploitation d'un système

[SOURCE: IEC 61499-1:2012, 3.92, modifié – Dans la définition, le terme "configurations" a été supprimé.]

3.2 Termes abrégés et acronymes

ASCII	American Standard Code for Information Interchange (Code normalisé américain pour l'échange d'informations) conformément à l'ISO/IEC 10646
CP	Communication Profile (Profil de communication)
CPF	Communication Profile Family (Famille de profils de communication)
CS	Communication Server (Serveur de communication)
EDD	Electronic Device Description (Description électronique de produit)

EDDL	Electronic Device Description Language (Langage de description électronique de produit)
EUC	Extended Unicode (Unicode étendu) conformément à l'ISO/IEC 10646
FB	Function Block (Bloc fonctionnel)
FF	FOUNDATION Fieldbus
GPE	Generic Protocol Extension (Extension du protocole générique)
HART	Highway Addressable Remote Transducer (Transducteur de canal adressable distant)
IHM	Interface homme-machine
E/S	Entrée/Sortie
ID	Identificateur
ISA100	ISA100_Wireless
OSI	Open Systems Interconnection (Interconnexion de systèmes ouverts)
PI	PROFIBUS et PROFINET International
PB	PROFIBUS
PDU	Protocol Data Unit (Unité de données de protocole)
PN	PROFINET
PNO	PROFIBUS Nutzerorganisation
QM	Quality Management (Management de la qualité)

4 Déclaration de conformité

Les déclarations de conformité à un profil du présent document doivent être déclarées comme

– conformes à l'IEC 61804-3 <Type>

où le Type entre les crochets obliques < > est facultatif et les crochets obliques ne doivent pas être inclus.

Le Type peut être, par exemple, l'abréviation du nom du protocole de communication utilisé, tel que CS, FF, GPE, HART, ISA100, PB ou PN, mais aussi être codé conformément aux profils de communication spécifiés dans l'IEC 61784-1 ou dans l'IEC 61784-2.

Les normes de produit ne doivent pas comprendre d'aspects relatifs à l'évaluation de conformité (dispositions QM incluses), normatifs ou informatifs, autres que les dispositions concernant la soumission à l'essai du produit (évaluation et examen).

5 Modèle EDD et EDDL

5.1 Présentation des EDD et du langage EDDL

Une description électronique de produit (EDD) contient tous les paramètres d'équipement d'un composant de système d'automatisation. 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 de base évolutifs pour gérer des équipements simples, complexes ou modulaires. Le langage EDDL est un langage descriptif fondé 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 multioctets.

5.2 Architecture EDD

Du point de vue du modèle 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 qui assure la prise en charge.

Le fabricant de l'équipement définit les objets, qui forment la représentation logique des objets d'une application EDD. Pour cette raison, EDDL comprend des éléments de langage, qui associent par mapping 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) d'un objet d'équipement.

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 EDD ou EDDL.

5.3 Concepts d'EDD

Le fabricant d'un équipement ou d'un composant d'un système d'automatisation décrit les propriétés de l'équipement en utilisant l'EDDL. L'EDD qui en résulte contient des informations telles que:

- la description des paramètres de l'équipement;
- la description des dépendances des paramètres;
- le regroupement logique des paramètres de l'équipement;
- la sélection et l'exécution des fonctions de l'équipement prises en charge;
- la logique métier (fonctions programmées);
- la description des ensembles de données transférés.

Suivant l'utilisation exigée, l'EDD peut être physiquement située

- dans un équipement;
- dans un support de stockage de données externe tel qu'un disque compact, une disquette ou un serveur;
- à la fois dans l'équipement 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 mise en œuvre EDD comporte suffisamment d'informations sur l'équipement cible pour faire correspondre une EDD spécifique avec un équipement spécifique. Le fabricant, le type d'équipement et la révision sont des exemples d'informations pouvant être utilisées pour mettre en correspondance un équipement et une EDD spécifique. Un exemple d'EDD est donné, à titre de référence, dans l'Annexe C.

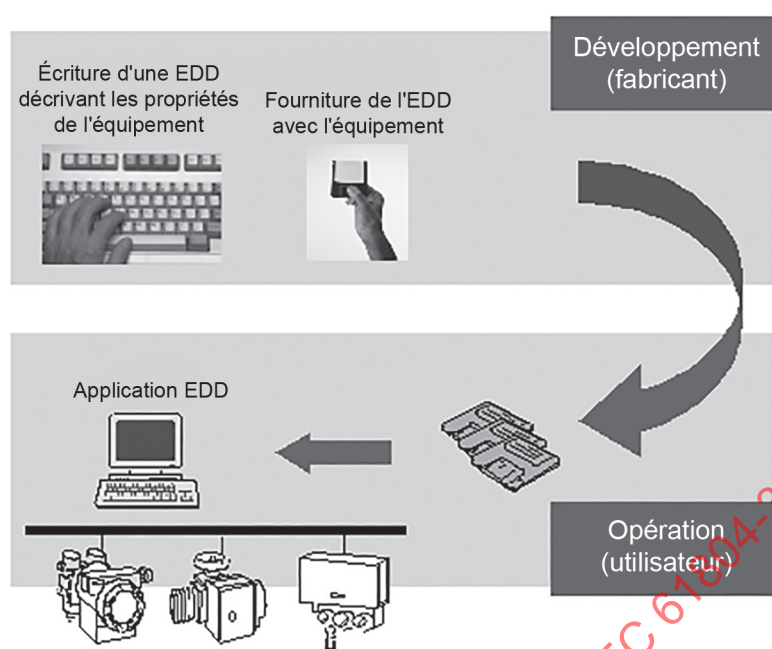
5.4 Principes du processus de développement d'EDD

5.4.1 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.

5.4.2 Génération du code source de l'EDD

Le processus de génération d'EDD est représenté à la Figure 2.



IEC

Figure 2 – Processus de génération d'EDD

Le fabricant d'un équipement écrit une EDD appropriée pour son équipement et fournit les deux (l'EDD et l'équipement). Si le système d'automatisation prend en charge la méthode EDD, une intégration de système peut être effectuée par l'utilisateur.

Les EDD pour les équipements peuvent être imbriquées dans la mémoire de l'équipement 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.

5.4.3 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.

5.4.4 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.

5.5 Interrelations entre la structure lexicale et les définitions formelles

La structure lexicale de l'EDDL et ses éléments sont décrits aux Articles 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 et 20. 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 à l'Annexe A, une autre spécification peut être développée en tant qu'option additionnelle.

5.6 Builtins

Les builtins sont des sous-programmes prédéfinis qui sont exécutés dans l'application EDD.

La bibliothèque de builtin est définie dans l'IEC 61804-5.

5.7 Profils

L'EDDL est une spécification unifiée de concepts EDD existants. Les applications EDD concrètes utilisent un sous-ensemble de la spécification EDDL. Les éléments de la spécification applicables aux différents profils sont définis dans les tableaux du présent document et accompagnés d'une description dudit élément.

En plus des profils de l'EDD, les consortiums implémenteurs publient également des "profils d'équipement", qui sont utilisés pour prendre en charge l'interchangeabilité d'équipements conformes. Ces profils d'équipements peuvent être décrits en utilisant l'EDDL spécifié dans le présent document.

6 Conventions

6.1 Généralités

Dans l'IEC 61804 (toutes les parties), 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.

6.2 Convention de description des constructions de base

Les conventions suivantes s'appliquent à la définition des constructions de base.

- Le Tableau 1 définit le format du tableau de présentation des builtins.
- Le Tableau 2 définit le contenu des colonnes du Tableau 1.
- Le Tableau 3 définit l'utilisation des attributs et sous-attributs des constructions de base.
- Les constructions de base EDDL et leurs attributs référencés dans le présent document sont écrits en majuscules.

Tableau 1 – Format du tableau des constructions de base

Référence	Construction de base	Attribut	Profils de communication

Tableau 2 – Contenu du tableau des constructions de base

Colonne	Texte	Signification
Référence		Référence au paragraphe correspondant, lequel définit l'attribut de la construction de base.
Construction de base		Nom de la construction de base.
Profils de communication	✓	La construction de base est prise en charge par le profil de communication correspondant.
	x	La construction de base n'est pas prise en charge par le profil de communication correspondant.

Tableau 3 – Description de l'utilisation des attributs

Utilisation	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 remplie.
L	Vvv	La présence de cet attribut est limitée et son utilisation est réservée à des cas spécifiques.

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 il n'est pas nécessaire qu'il soit présent;
- s: cet attribut est un choix. Un et un seul des champs marqués par "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.

6.3 Conventions pour la structure lexicale

Les structures lexicales du présent document sont spécifiées à l'aide des règles définies dans le Tableau 4.

Tableau 4 – Règles de convention lexicale

Elément de structure lexicale	Description
gras	Les littéraux syntaxiques concrets sont en gras .
<i>italique</i>	Les éléments syntaxiques sont en <i>italique</i> .
:=	Définition d'un élément syntaxique.
<cond>	L'élément syntaxique précédent peut-être défini par une condition.
[]	Groupe d'éléments syntaxiques.
*	L'élément syntaxique précédent peut apparaître zéro ou plusieurs fois. L'élément syntaxique peut être un groupe d'éléments syntaxiques placés entre crochets.
+	L'élément syntaxique précédent peut apparaître une ou plusieurs fois. L'élément syntaxique peut être un groupe d'éléments syntaxiques placés entre crochets.
?	L'élément syntaxique précédent peut apparaître zéro ou une fois. L'élément syntaxique peut être un groupe d'éléments syntaxiques placés entre crochets.
	Eléments syntaxiques alternatifs.
expression	Expression.
<type de données>-expression	Expression pouvant être résolue par le type de données spécifié.
<type de données>-constante	Valeur constante du type de données spécifié.
<type de données>-valeur	Valeur du type de données spécifié.

Le Tableau 5 décrit les sources EDD applicables aux définitions des éléments de structure lexicale.

Tableau 5 – Sources EDD

Elément de structure lexicale	Source applicable						
	Constante (littéral)	Attribut référencé	Valeur énumérée()	Référence de dictionnaire (chaîne uniquement)	Concaténation ("+", chaîne uniquement)	Expression (conversion de type incluse)	Conditions (IF, SELECT)
<type de données>-constante	x	x	x	x			
<type de données>-valeur	x	x	x	x	x		
<type de données>-valeur <cond>	x	x	x	x	x		x
<type de données>-expression	x	x	x	x	x	x	
<type de données>-expression <cond>	x	x	x	x	x	x	x

7 Vue d'ensemble

7.1 Caractéristiques du langage EDDL

L'EDDL est un langage structuré et interprétatif pour décrire les propriétés d'un équipement. De plus, les interactions entre les équipements 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 mise en œuvre EDD spécifique, il n'est pas nécessaire d'utiliser tous les éléments fournis par le langage.

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 développeurs d'EDD sont tenus d'identifier dans chaque EDD les informations sur le profil qui a été utilisé.

7.2 Représentation de syntaxe

La structure lexicale 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 le présent document, il est possible d'utiliser d'autres définitions de syntaxe, qui peuvent être ajoutées à l'avenir pour permettre d'autres caractéristiques et représentations pour une évolution future.

7.3 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 de séparer les chaînes de texte fréquemment utilisées (un texte d'aide et des listes multilingues de libellés, par exemple) dans un dictionnaire de texte (voir Article 20).

Les informations d'identification sont la première entrée de chaque fichier EDD. Les informations d'identification identifient de manière unique la version du langage EDDL utilisée, conjointement avec le type d'équipement, les codes du modèle et les éléments de révision spécifiques couverts par le fichier EDD.

7.4 Éléments de construction de base

7.4.1 Généralités

Ces constructions de base ont été spécifiées pour prendre en charge les descriptions d'équipements 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 précises. Cette variété additionnelle a été incluse afin d'assurer la compatibilité avec plusieurs langages de description existants. Des références croisées de mappings appropriés sont données par les profils.

Chacune des constructions de base a un jeu d'attributs associé. Les attributs peuvent également 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.

7.4.2 AXIS

AXIS décrit l'axe d'un attribut CHART ou GRAPH (voir 11.7).

7.4.3 BLOB

BLOB (Binary Large Object) est un grand objet binaire utilisé pour transférer des données binaires depuis ou vers un équipement (voir 9.8).

7.4.4 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 13.1 et Figure 3).

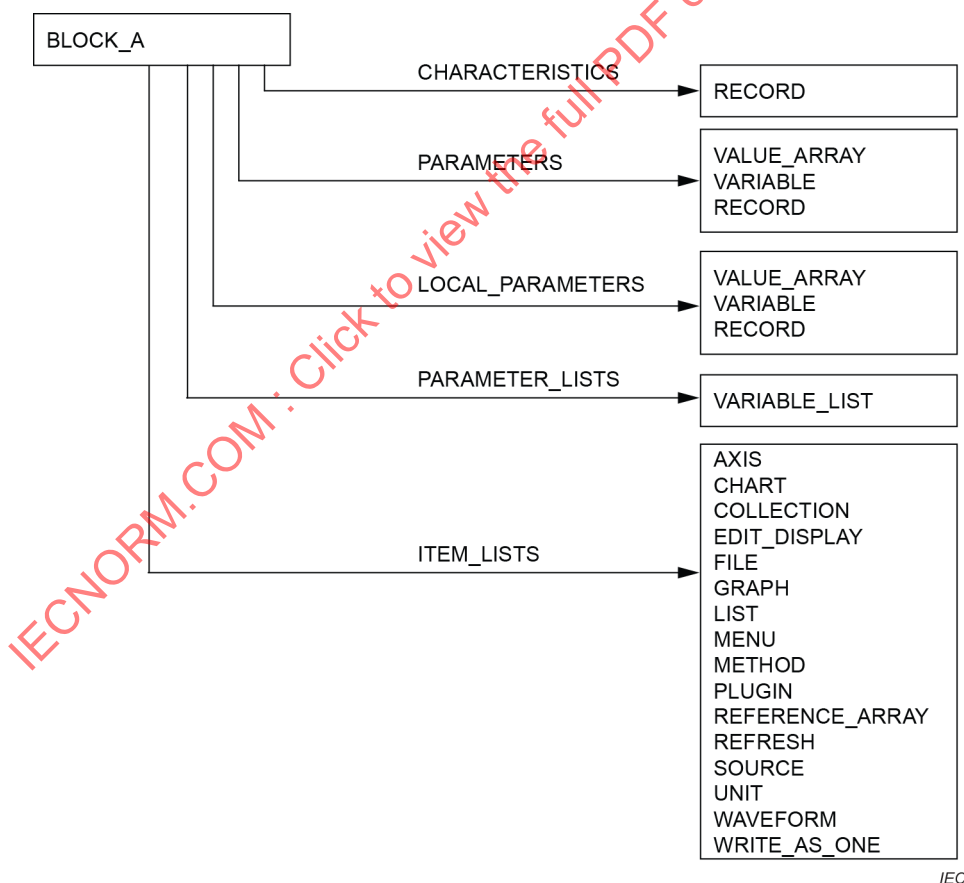


Figure 3 – BLOCK_A

7.4.5 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 permettre un adressage relatif (voir 13.2).

7.4.6 CHART

CHART décrit un diagramme utilisé pour afficher des données provenant d'un équipement dans l'application EDD (voir 11.5 et Figure 4).



Figure 4 – CHART

7.4.7 COLLECTION

COLLECTION décrit des regroupements logiques de données (voir 9.5 et Figure 5).

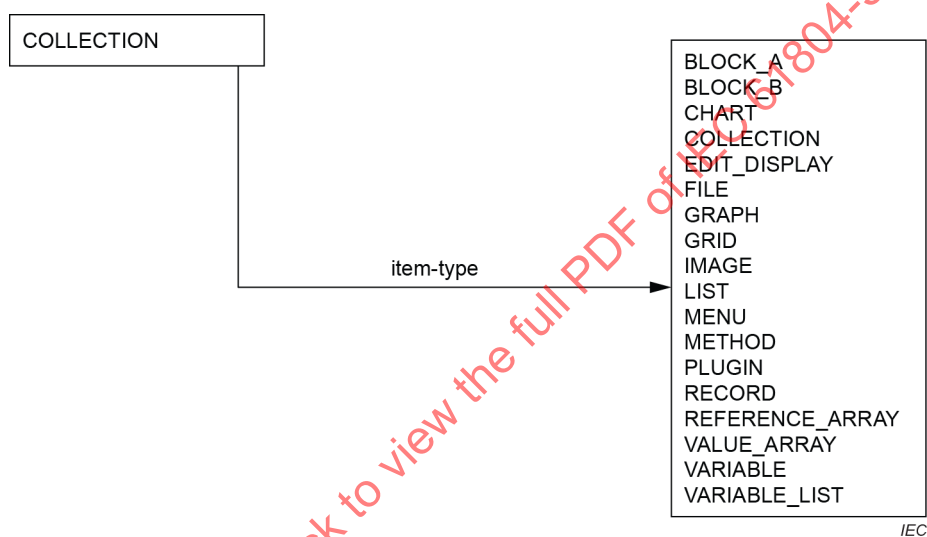


Figure 5 – COLLECTION

7.4.8 COMMAND

COMMAND décrit la structure et l'adressage des variables dans l'équipement (voir 10.1 et Figure 6).

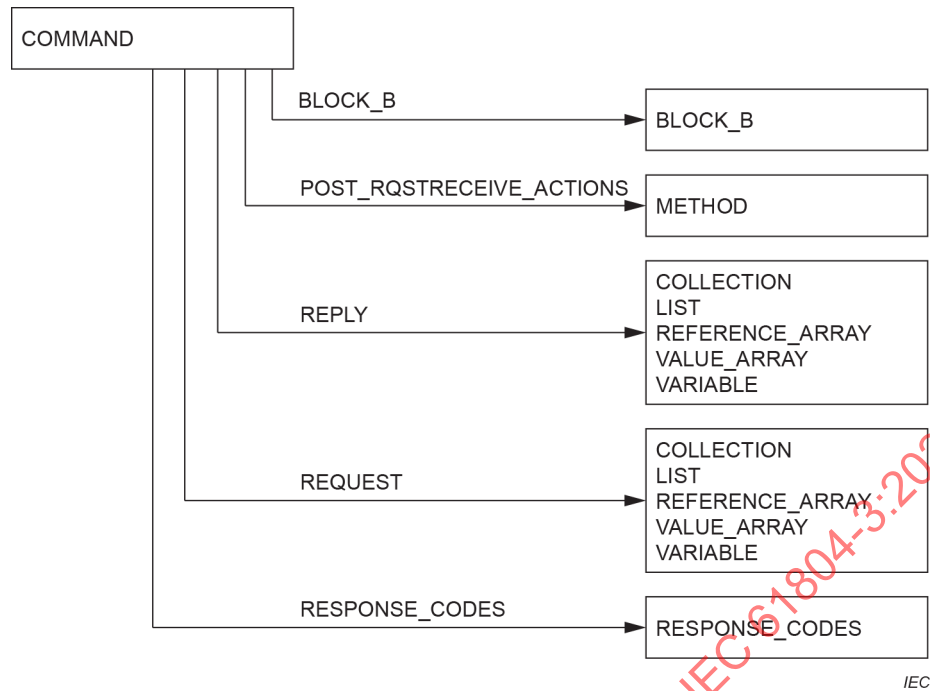


Figure 6 – COMMAND

7.4.9 COMPONENT

COMPONENT décrit le comportement de configuration d'un équipement qui est décrit dans l'EDD (voir 13.6 et Figure 7).

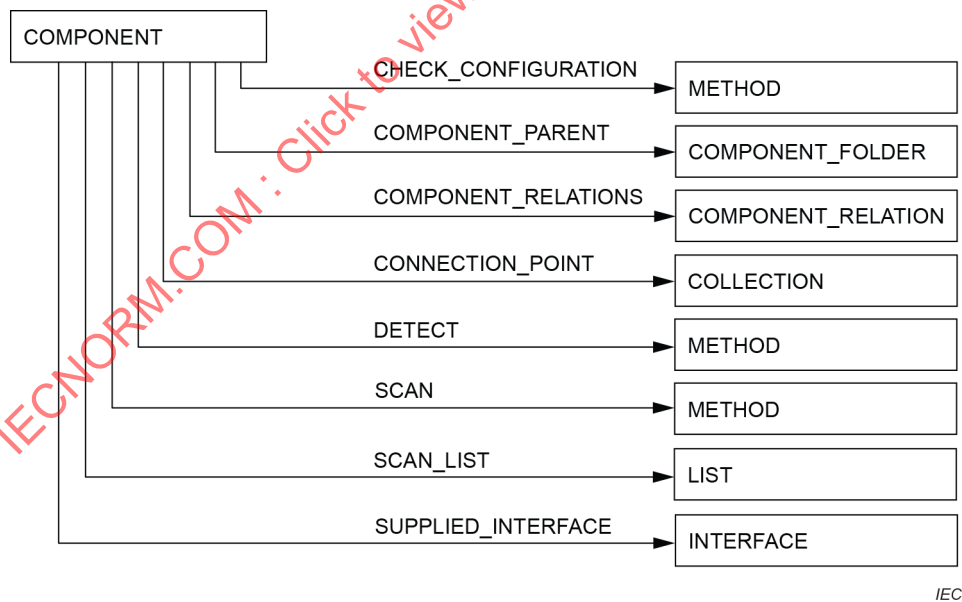


Figure 7 – COMPONENT

7.4.10 COMPONENT_FOLDER

COMPONENT_FOLDER décrit un dossier dans un catalogue. Les dossiers permettent de regrouper des composants dans un catalogue (voir 13.7 et Figure 8).



Figure 8 – COMPONENT_FOLDER

7.4.11 COMPONENT_REFERENCE

COMPONENT_REFERENCE fait référence à un attribut COMPONENT ou COMPONENT_FOLDER (voir 13.8 et Figure 9).



Figure 9 – COMPONENT_REFERENCE

7.4.12 COMPONENT_RELATION

COMPONENT_RELATION spécifie une relation entre des attributs COMPONENT, par exemple un module et un ou plusieurs autres modules (voir 13.9 et Figure 10).

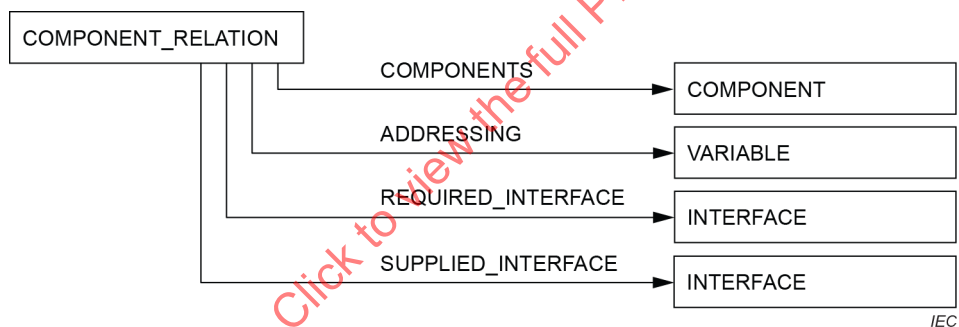


Figure 10 – COMPONENT_RELATION

7.4.13 EDIT_DISPLAY

EDIT_DISPLAY décrit la manière dont les données sont présentées à un utilisateur par un équipement d'affichage (voir 11.2 et Figure 11).

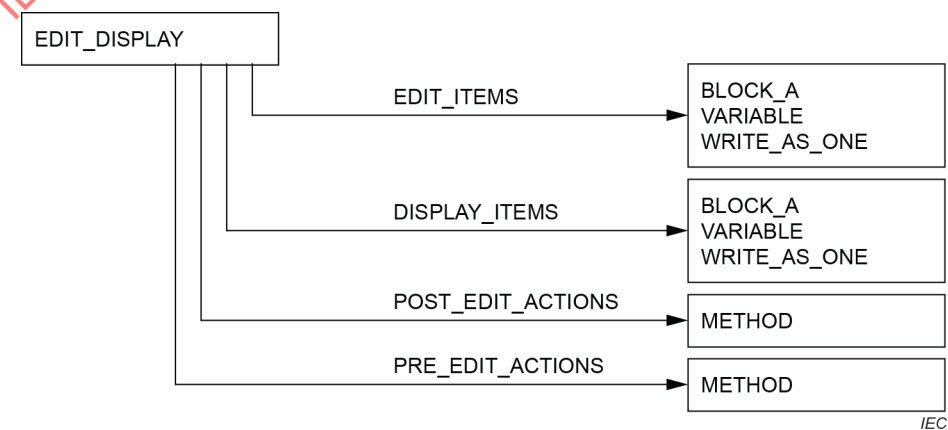


Figure 11 – EDIT_DISPLAY

7.4.14 FILE

FILE décrit une zone de stockage persistant (voir 9.6 et Figure 12).

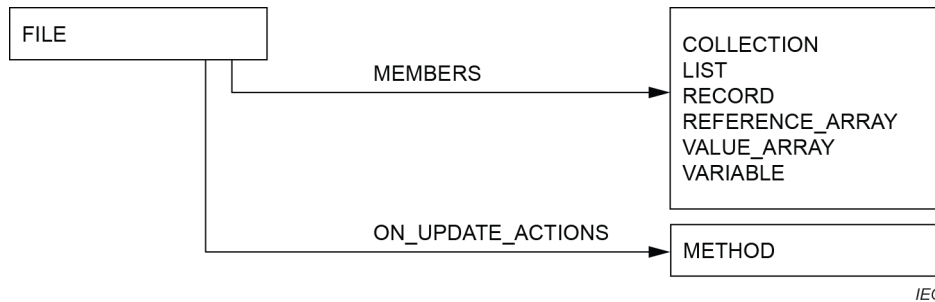


Figure 12 – FILE

7.4.15 GRAPH

GRAPH décrit un graphique utilisé pour afficher des données provenant d'un équipement dans l'application EDD (voir 11.8 et Figure 13).

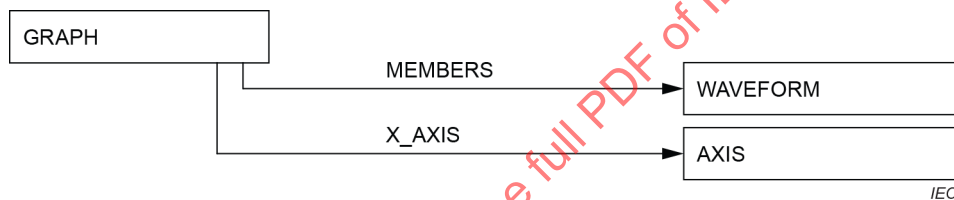


Figure 13 – GRAPH

7.4.16 GRID

GRID décrit une matrice utilisée pour afficher des données provenant d'un équipement dans l'application EDD (voir 11.4 et Figure 14).



Figure 14 – GRID

7.4.17 IMAGE

IMAGE décrit une image visuelle (voir 11.3 et Figure 15).



Figure 15 – IMAGE

7.4.18 IMPORT

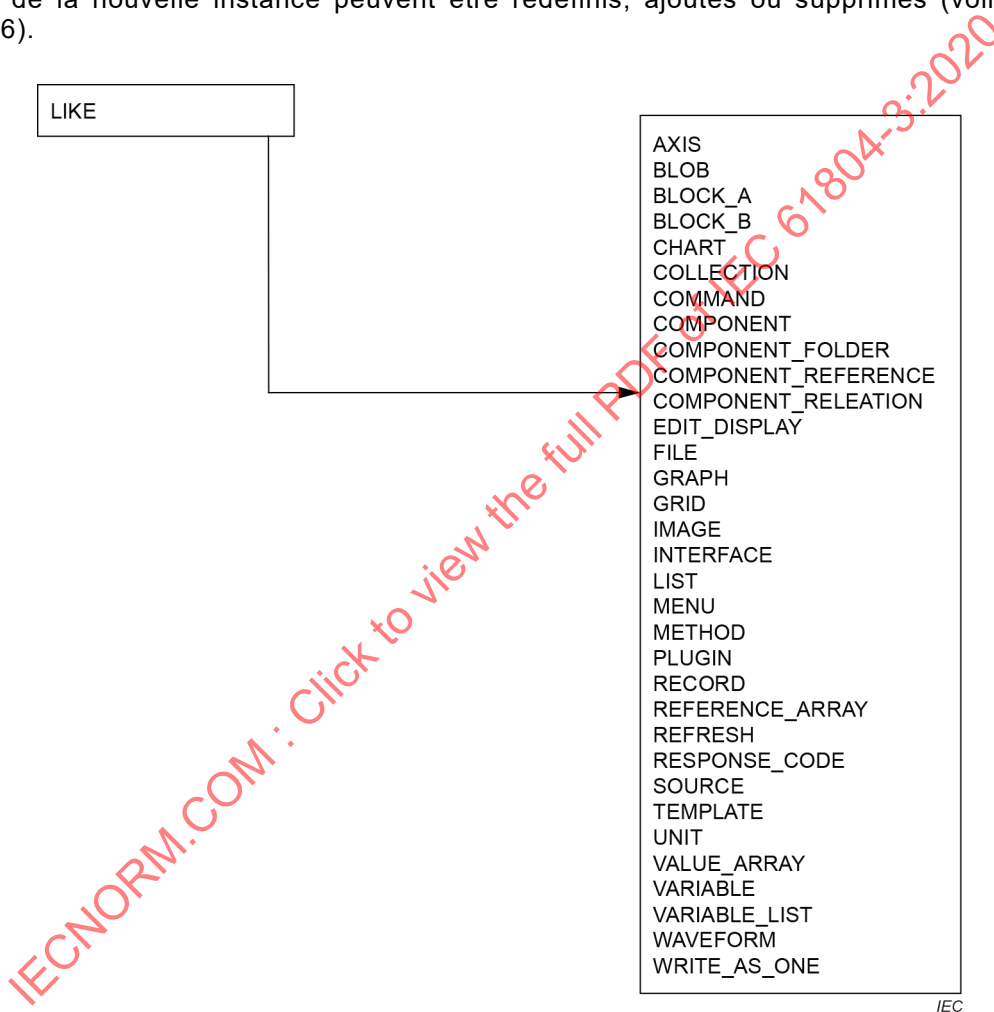
IMPORT est utilisé pour importer une EDD et modifier ses définitions existantes (voir 13.4).

7.4.19 INTERFACE

INTERFACE est utilisé pour spécifier un ensemble de constructions de base de composants qui sont accessibles par d'autres composants (voir 13.10).

7.4.20 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 13.3 et Figure 16).



IEC

Figure 16 – LIKE

7.4.21 LIST

LIST décrit une séquence d'éléments (voir 9.4 et Figure 17).

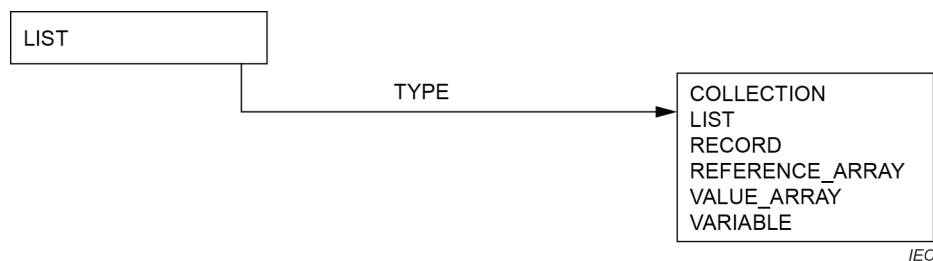


Figure 17 – LIST

7.4.22 MENU

MENU décrit la manière dont les données sont présentées à un utilisateur par une application EDD (voir 11.1 et Figure 18).

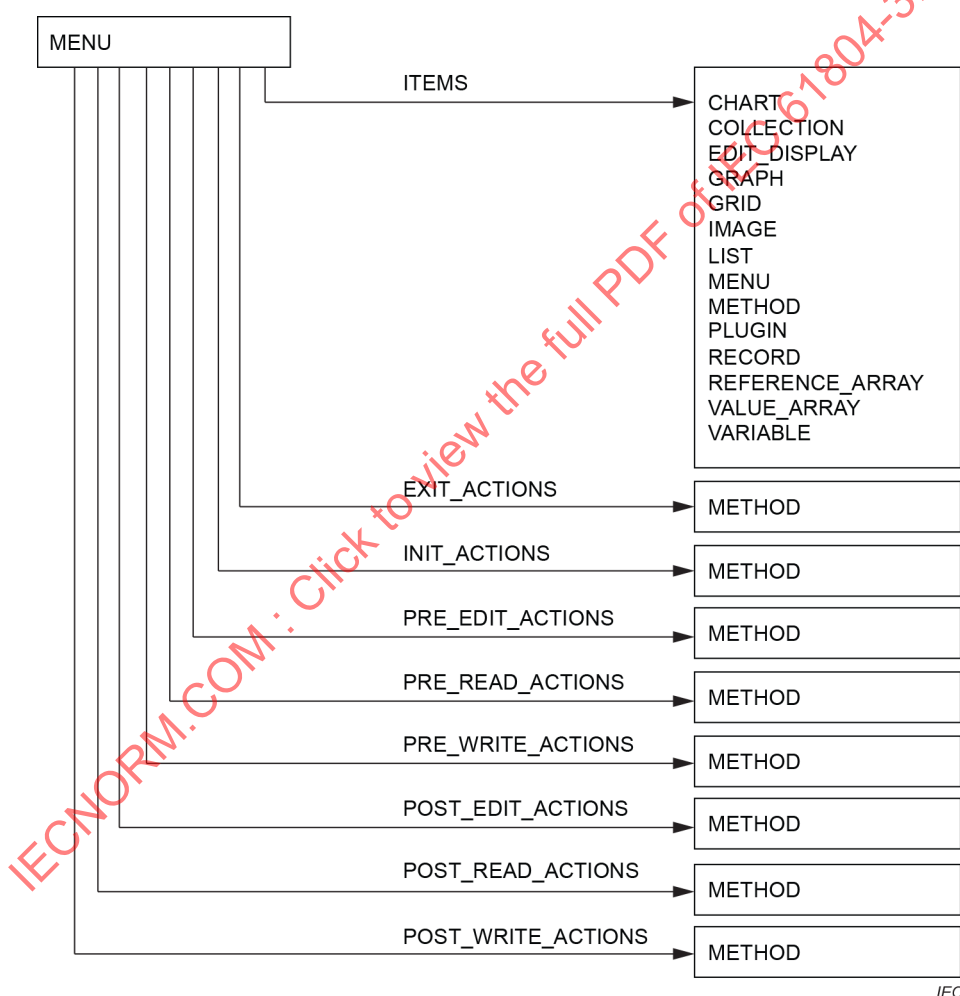


Figure 18 – MENU

7.4.23 METHOD

METHOD décrit l'exécution d'une séquence complexe d'interactions d'événements qui se produisent entre les équipements du système tels que des unités d'affichage, des configureurs et d'autres équipements (voir Article 12).

7.4.24 PLUGIN

PLUGIN décrit un plug-in d'interface utilisateur (UIP, *User Interface Plug-in*), spécifié en 11.10.

7.4.25 RECORD

RECORD décrit les données contenues dans l'équipement (voir 10.2 et Figure 19).

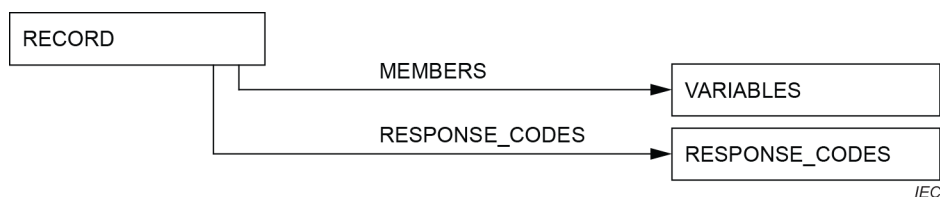


Figure 19 – RECORD

7.4.26 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 par le nom de REFERENCE_ARRAY et l'index associé à l'élément (voir 9.2 et Figure 20).

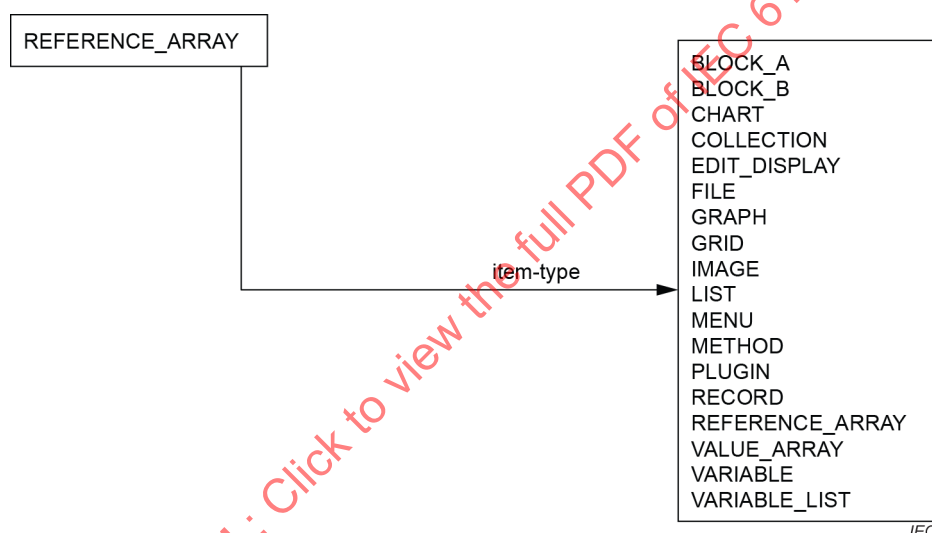


Figure 20 – REFERENCE_ARRAY

7.4.27 Relations

7.4.27.1 REFRESH

REFRESH décrit les relations entre les attributs LIST, RECORD, VALUE_ARRAY et VARIABLE (voir 9.7.1 et Figure 21).

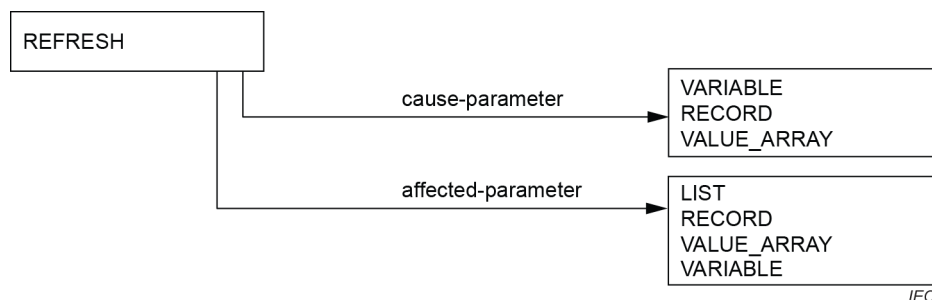


Figure 21 – REFRESH

7.4.27.2 SEMANTIC_MAP

SEMANTIC_MAP décrit les relations entre les identificateurs sémantiques normalisés et les éléments de données (voir 9.7.2 et Figure 22). Les identificateurs sémantiques sont des chaînes qui sont indépendantes des paramètres régionaux et sont généralement spécifiées dans les Normes internationales comme le Dictionnaire de données commun (CDD, *Common Data Dictionary*) de l'IEC fondé sur l'IEC 61360 (toutes les parties) et l'IEC 61987 (toutes les parties).

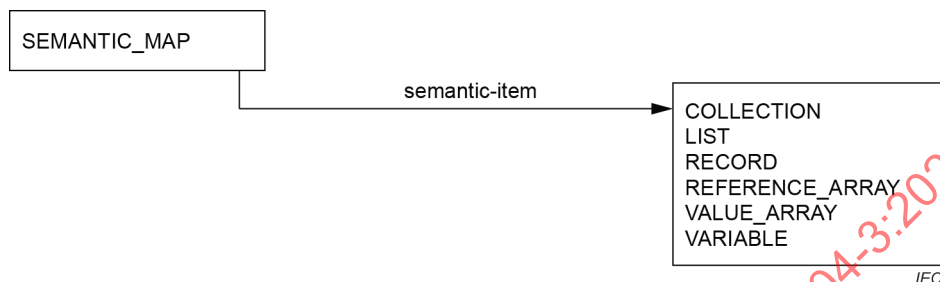


Figure 22 – SEMANTIC_MAP

7.4.27.3 UNIT

UNIT décrit les relations entre les attributs AXIS, VALUE_ARRAY et VARIABLE (voir 9.7.3 et Figure 23).

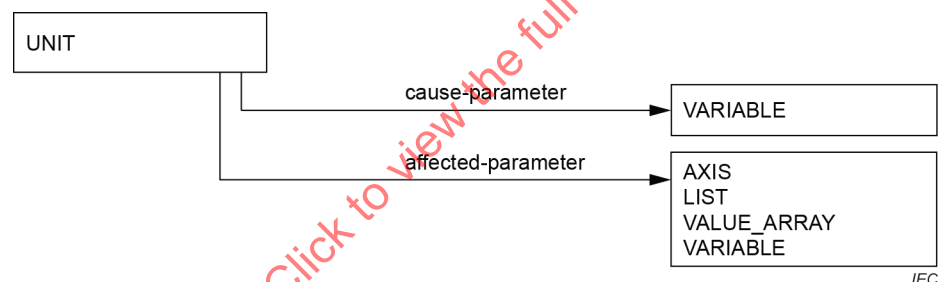


Figure 23 – UNIT

7.4.27.4 WRITE_AS_ONE

WRITE_AS_ONE décrit les relations entre les attributs VARIABLE, RECORD et VALUE_ARRAY (voir 9.7.4 et Figure 24).



Figure 24 – WRITE_AS_ONE

7.4.28 RESPONSE_CODES

RESPONSE_CODES spécifie la liste des éléments de code de réponse utilisés par l'application EDD pour informer l'utilisateur du résultat de la communication, par exemple erreur, avertissement (voir 10.3).

7.4.29 SOURCE

SOURCE décrit une source de valeurs de données pour un attribut CHART (voir 11.6 et Figure 25).

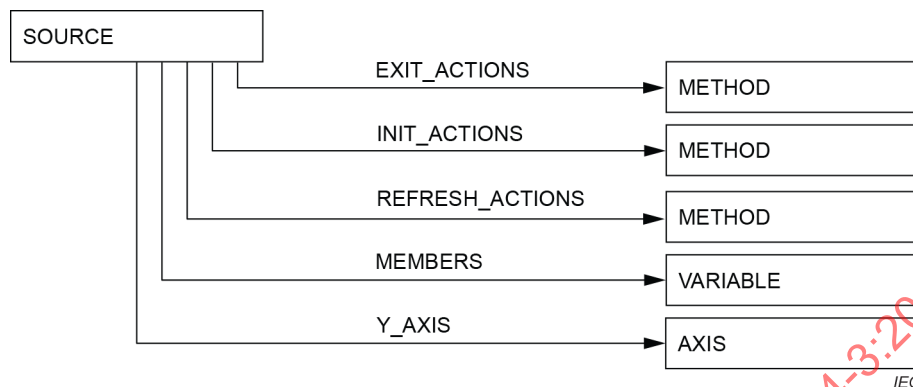


Figure 25 – SOURCE

7.4.30 TEMPLATE

TEMPLATE décrit un ensemble de valeurs de paramètres utilisé pour la configuration hors ligne par l'application EDD (voir 13.5).

7.4.31 VALUE_ARRAY

VALUE_ARRAY décrit des groupes logiques de variables EDDL (voir 9.3 et Figure 26).

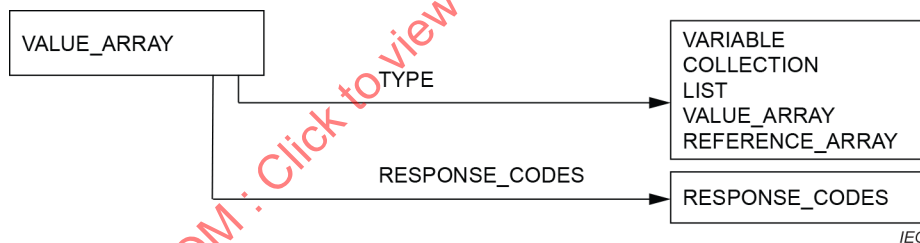


Figure 26 – VALUE_ARRAY

7.4.32 VARIABLE

VARIABLE décrit les données contenues dans l'équipement (voir 9.1 et Figure 27).

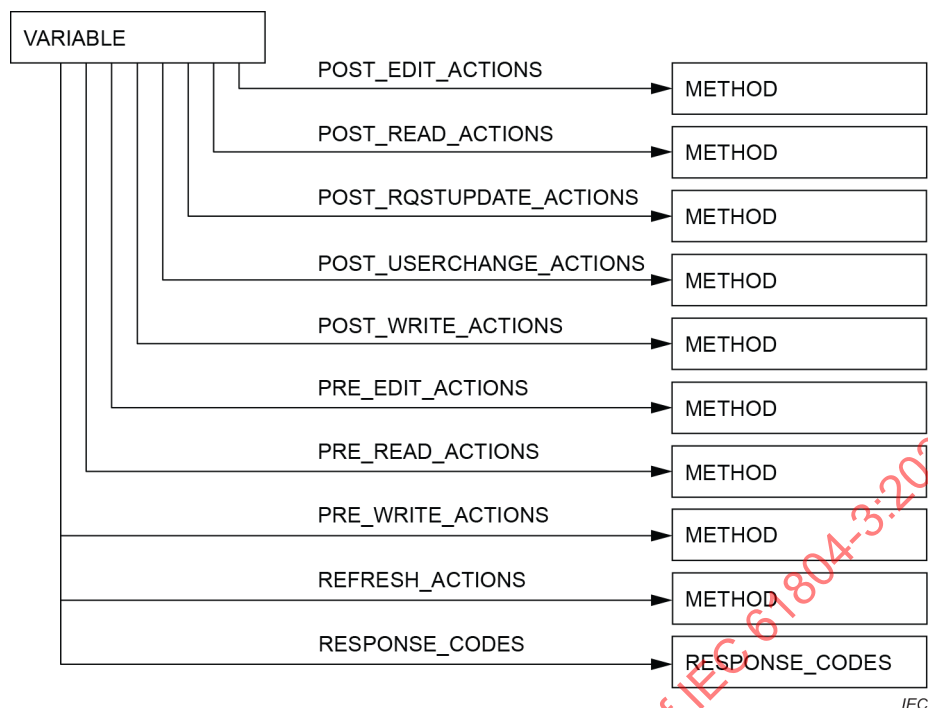


Figure 27 – VARIABLE

7.4.33 VARIABLE_LIST

VARIABLE_LIST décrit les regroupements logiques de données contenues dans l'équipement qui peuvent être communiqués sous forme de liste (voir 10.4 et Figure 28).

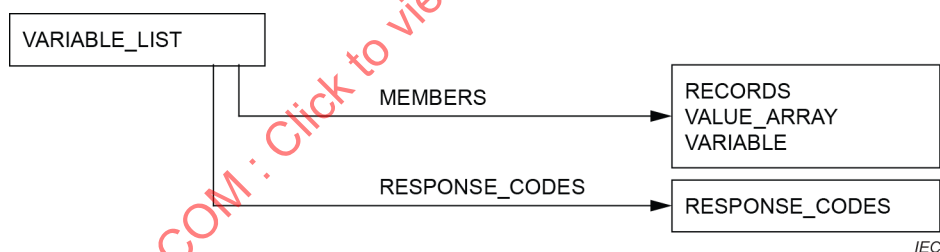


Figure 28 – VARIABLE_LIST

7.4.34 WAVEFORM

WAVEFORM décrit un ensemble de données qui peut être affiché par un attribut GRAPH (voir 11.9 et Figure 29).

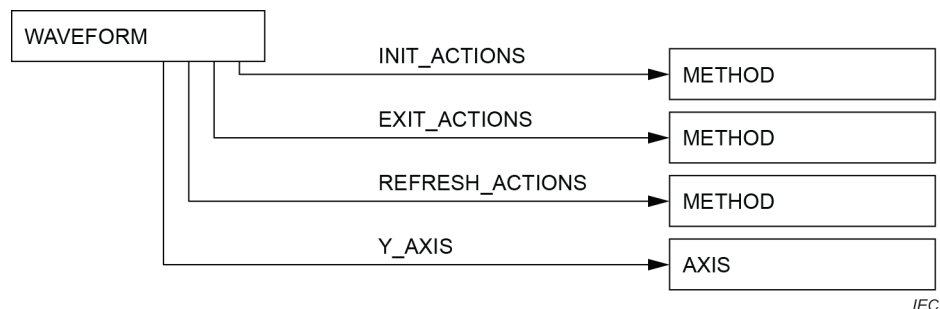


Figure 29 – WAVEFORM

7.5 Attributs communs

Les attributs communs suivants sont utilisés dans de nombreuses constructions de base:

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

7.6 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, comme:

- 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'équipement 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.

7.7 Règles pour les instances

Chaque instance d'une construction de base est identifiée par un identificateur de type chaîne. Cet identificateur est unique dans l'ensemble de l'EDD. Toutes les constructions de l'EDDL représentent des instances réelles d'informations de l'équipement. Si un équipement contient deux éléments d'information ou plus avec la même structure, la construction appropriée est répétée dans l'EDD avec des identificateurs différents. En variante, la construction 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.

7.8 Règles pour une liste d'attributs VARIABLE

Une liste d'attributs VARIABLE peut contenir des VARIABLES, 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.

8 Informations d'identification d'une EDD

8.1 Structure générale

Objet

Les informations d'identification d'une EDD identifient de manière unique la description de l'équipement d'un type d'équipement spécifique d'un fabricant d'équipements. Les applications EDD peuvent lire les paramètres de l'équipement qui correspondent à ces attributs et les utiliser pour identifier l'EDD. Les informations d'identification d'une EDD 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 integer-constant ,
DEVICE_REVISION integer-constant ,
DEVICE_TYPE [integer-constant | identifier] ,
EDD_PROFILE integer-constant ,
EDD_VERSION integer-constant ,
LAYOUT_TYPE [COLUMNWIDTH_EQUAL | COLUMNWIDTH_OPTIMIZED] ,
MANUFACTURER [integer-constant | identifier] ,
MANUFACTURER_EXT string-constant
```

Les attributs d'identification d'EDD sont spécifiés dans le Tableau 6.

Tableau 6 – Attributs d'informations d'identification d'EDD

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
8.2	DD_REVISION	m	m	m	m	m	m	m
8.3	DEVICE_REVISION	m	m	m	m	m	m	m
8.4	DEVICE_TYPE	m	m	m	m	m	m	m
8.5	EDD_PROFILE	o	x	o	x	o	o	o
8.6	EDD_VERSION	o	x	o	x	x	o	o
8.7	LAYOUT_TYPE	o	o	o	o	o	o	o
8.8	MANUFACTURER	m	m	m	m	m	m	m
8.9	MANUFACTURER_EXT	c	x	c	x	x	c	c

8.2 DD_REVISION

Objet

L'attribut DD_REVISION spécifie un code unique pour la révision de cette EDD de l'équipement, défini par le fabricant.

Il peut exister plusieurs EDD pour une révision particulière d'un équipement particulier, auquel cas l'attribut DD_REVISION permet de les distinguer. Toutefois, une EDD individuelle ne peut pas prendre en charge plus d'un type ou d'une révision d'un équipement.

Il convient que le fabricant modifie le numéro de révision chaque fois qu'une nouvelle description d'équipement est émise pour un attribut DEVICE_REVISION spécifique de l'équipement. En outre, les règles de révision DD peuvent être conditionnées par les politiques établies par les fondations de protocoles.

Structure lexicale

DD_REVISION *integer-constant*

L'attribut DD_REVISION est spécifié dans le Tableau 7.

Tableau 7 – Attribut DD_REVISION

Attribut	Description
integer-constant	Profils de communication: CS, GPE, HART, PB, PN Entier non signé à deux octets qui identifie de manière unique le niveau de révision de cette description d'équipement pour l'équipement, défini par le fabricant. Profils de communication: FF, ISA100 Entier non signé à un octet qui identifie de manière unique le niveau de révision de cette description d'équipement pour l'équipement, défini par le fabricant.

8.3 DEVICE_REVISION

Objet

L'attribut DEVICE_REVISION spécifie un code unique pour la révision de l'équipement, défini par le fabricant. Les règles de révision de l'équipement peuvent être conditionnées par les politiques établies par les fondations de protocoles.

Structure lexicale

DEVICE_REVISION *integer-constant*

L'attribut DEVICE_REVISION est spécifié dans le Tableau 8.

Tableau 8 – Attribut DEVICE_REVISION

Attribut	Description
integer-constant	Profils de communication: CS, GPE, HART, PB, PN, ISA100 Entier non signé à deux octets qui identifie de manière unique le niveau de révision de l'équipement, défini par le fabricant.
integer-constant	Profils de communication: FF Entier non signé à un octet qui identifie de manière unique le niveau de révision de l'équipement, défini par le fabricant.

8.4 DEVICE_TYPE

Objet

L'attribut DEVICE_TYPE spécifie un identificateur pour le type d'équipement, défini par le fabricant de l'équipement. Il convient que l'identificateur soit unique pour chaque type relevant d'un même fabricant.

Le type d'équipement peut spécifier une catégorie d'équipements ou un produit spécifique.

Structure lexicale

DEVICE_TYPE [*integer-constant* | *identifier*]

Les attributs DEVICE_TYPE sont spécifiés dans le Tableau 9.

Tableau 9 – Attribut DEVICE_TYPE

Attribut	Description
integer-constant	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Entier non signé à deux octets contenant le code pour un type d'équipement ou groupe de types d'équipements spécifique.
identifier	Profils de communication: CS, GPE, HART, PB, PN Référence à un entier à deux octets qui contient le code pour un type d'équipement ou groupe de types d'équipements spécifique.
Le mapping de l'identificateur vers une constante entière peut être stocké dans un fichier séparé appelé "devices.cfg". Ce fichier contient une liste d'identificateurs pour des équipements ou des groupes de types d'équipements.	

8.5 EDD_PROFILE

Objet

L'attribut EDD_PROFILE spécifie le profil sélectionné d'EDDL, défini dans l'Annexe B.

Structure lexicale

EDD_PROFILE *integer-constant*

L'attribut EDD_PROFILE est spécifié dans le Tableau 10.

Tableau 10 – Attribut EDD_PROFILE

Attribut	Description
integer-constant	Entier non signé à un octet contenant le code pour les profils spécifiques. Les codes suivants sont actuellement définis: 0x01 Profil pour HART 0x02 Profil pour FOUNDATION Fieldbus 0x03 Profil pour PROFIBUS 0x04 Profil pour PROFINET 0x05 Profil pour ISA100_Wireless 0x06 Profil pour GPE 0x07 Profil pour CS

NOTE Même si les profils de communication ne prennent pas tous en charge l'attribut EDD_PROFILE, les valeurs assignées à tous les profils sont répertoriées dans le Tableau 10 à des fins d'exhaustivité.

8.6 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'équipement conformes au présent document.

Structure lexicale

EDD_VERSION *integer-constant*

L'attribut EDD_VERSION est spécifié dans le Tableau 11.

Tableau 11 – Attribut EDD_VERSION

Attribut	Description
integer	Entier non signé à deux octets contenant le chiffre 1.

8.7 LAYOUT_TYPE

Objet

L'attribut LAYOUT_TYPE spécifie comment l'application EDD doit restituer l'intégralité de l'interface utilisateur graphique définie dans cette EDD.

Structure lexicale

LAYOUT_TYPE [COLUMNWIDTH_EQUAL | COLUMNWIDTH_OPTIMIZED]

Les attributs LAYOUT_TYPE sont spécifiés dans le Tableau 12.

Tableau 12 – Attribut LAYOUT_TYPE

Valeur d'attribut	Description
COLUMNWIDTH_EQUAL	Tous les MENUS dont l'attribut STYLE est WINDOW, DIALOG, PAGE ou GROUP doivent être restitués avec des colonnes de largeur égale. Profils de communication: FF, HART, ISA100 La valeur par défaut de LAYOUT_TYPE est COLUMNWIDTH_EQUAL
COLUMNWIDTH_OPTIMIZED	Tous les MENUS dont l'attribut STYLE est WINDOW, DIALOG, PAGE ou GROUP doivent être restitués avec des colonnes de largeur optimisée. Les colonnes de largeur optimisée doivent être dimensionnées par rapport à l'élément le plus large de la colonne pour toutes les lignes contenant le même nombre de colonnes. Profils de communication: CS, GPE, PB, PN La valeur par défaut de LAYOUT_TYPE est COLUMNWIDTH_OPTIMIZED

8.8 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 [*integer-constant* | *identifieur*]

Les attributs MANUFACTURER sont spécifiés dans le Tableau 13.

Tableau 13 – Attribut MANUFACTURER

Attribut	Description
integer-constant	Profils de communication: CS, GPE, HART, PB, PN Entier non signé contenant le code pour un fabricant spécifique. Profils de communication: CS, GPE, PB, PN Si la valeur de l'entier est 0xFFFFF, la structure lexicale "MANUFACTURER_EXT" doit être présente et identifie le fabricant.
integer-constant	Profils de communication: FF Entier non signé de 24 bits contenant le code pour un fabricant spécifique.
integer-constant	Profils de communication: ISA100 Entier non signé de 32 bits contenant le code pour un fabricant spécifique. L'octet de poids fort de la valeur MANUFACTURER doit être défini sur 0.
identifieur	Profils de communication: CS, GPE, HART, PB, PN Référence à un entier long non signé qui contient le code pour un fabricant spécifique.
Il convient de stocker le mapping de l'identificateur vers une constante entière dans un fichier séparé appelé "devices.cfg". Ce fichier contient une liste d'identificateurs pour les fabricants.	

8.9 MANUFACTURER_EXT

Objet

L'attribut MANUFACTURER_EXT est 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-constant*

L'attribut MANUFACTURER_EXT est spécifié dans le Tableau 14.

Tableau 14 – Attribut MANUFACTURER_EXT

Attribut	Description
string-constant	Chaîne contenant les informations pour l'identification mondiale unique d'un fabricant.

9 Modélisation de données

9.1 VARIABLE

9.1.1 Structure générale

Objet

VARIABLE décrit des paramètres contenus dans un équipement ou dans l'application EDD. Ces éléments de données comprennent les données de processus, l'état et les propriétés relatives à la configuration. La construction VARIABLE fournit un large éventail de propriétés descriptives permettant de décrire intégralement tout élément de données.

Structure lexicale

```
VARIABLE identifier
{
    LABEL string-value;
    HELP string-value;
    CLASS variable-class-specifier [&variable-class-specifier]*;
    CONSTANT_UNIT string-value;
    DEFAULT_VALUE default-value-specifier;
    HANDLING handling-specifier;
    HEIGHT height-width-specifier;
    INITIAL_VALUE initial-value-specifier
    POST_EDIT_ACTIONS { actions-specifier }
    POST_READ_ACTIONS { actions-specifier }
    POST_RESETUPDATE_ACTIONS { actions-specifier }
    POST_USERCHANGE_ACTIONS { actions-specifier }
    POST_WRITE_ACTIONS { actions-specifier }
    PRE_EDIT_ACTIONS { actions-specifier }
    PRE_READ_ACTIONS { actions-specifier }
    PRE_WRITE_ACTIONS { actions-specifier }
    PRIVATE boolean-specifier;
    REFRESH_ACTIONS { actions-specifier }
    RESPONSE_CODES response-codes-specifier
    TYPE variable-type;
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
```

```

WIDTH height-width-specifier;
WRITE_MODE write-mode-specifier;
}

```

Les attributs VARIABLE sont spécifiés dans le Tableau 15.

Tableau 15 – Attributs VARIABLE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	VARIABLE		✓	✓	✓	✓	✓	✓	✓
9.1.2	CLASS		m	m	m	o	o	m	m
9.1.3	CONSTANT_UNIT		o	o	o	o	o	o	o
9.1.4	DEFAULT_VALUE		o	o	o	o	o	o	o
14.6	HANDLING		o	o	o	o	o	o	o
14.7	HEIGHT		o	o	o	o	o	o	o
14.8	HELP		o	o	o	o	o	o	o
9.1.5	INITIAL_VALUE		o	x	o	x	x	o	o
14.9	LABEL		o	m	o	o	m	o	o
9.1.6	POST_EDIT_ACTIONS		o	o	o	o	o	o	o
9.1.7	POST_READ_ACTIONS		o	o	o	o	o	o	o
9.1.8	POST_RQSTUPDATE_ACTIONS		x	x	x	o	x	x	x
9.1.9	POST_USERCHANGE_ACTIONS		x	x	x	o	x	x	x
9.1.10	POST_WRITE_ACTIONS		o	o	o	o	o	o	o
9.1.11	PRE_EDIT_ACTIONS		o	o	o	o	o	o	o
9.1.12	PRE_READ_ACTIONS		o	o	o	o	o	o	o
9.1.13	PRE_WRITE_ACTIONS		o	o	o	o	o	o	o
14.13	PRIVATE		o	o	o	o	o	o	o
9.1.14	REFRESH_ACTIONS		o	o	o	o	o	o	o
14.15	RESPONSE_CODES		x	o	x	x	o	x	x
9.1.15	TYPE		m	m	m	m	m	m	m
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	o	o	o	o
14.19	WIDTH		o	o	o	o	o	o	o
14.20	WRITE_MODE		x	o	x	x	o	x	x

9.1.2 CLASS

Objet

L'attribut CLASS spécifie comment l'attribut VARIABLE est utilisé par l'équipement et l'application EDD à des fins d'organisation et d'affichage. Les classes sont utilisées pour modéliser l'application de l'élément EDD (par exemple, indiquer que l'élément est utilisé pour diagnostiquer l'état de l'équipement). En outre, un élément EDD peut être fourni pour aider l'application EDD à accéder à l'équipement. Dans ce cas, la classe de l'élément EDD est LOCAL et celui-ci n'existe pas réellement dans l'équipement. L'attribut CLASS ne doit pas contenir d'éléments conditionnels.

La définition de CLASS est fondée sur un modèle dans lequel un équipement est constitué d'un certain nombre 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 *variable-class-specifier* [**&***variable-class-specifier*]*;
variable-class-specifier := **ALARM** | **ANALOG_INPUT** | **ANALOG_OUTPUT** |
COMPUTATION | **CONSTANT** | **CONTAINED** | **CORRECTION** | **DEVICE** | **DIAGNOSTIC** |
DIGITAL_INPUT | **DIGITAL_OUTPUT** | **DISCRETE** | **DISCRETE_INPUT** |
DISCRETE_OUTPUT | **DYNAMIC** | **FREQUENCY** | **FREQUENCY_INPUT** | **FREQUENCY_OUTPUT** |
HART | **INPUT** | **IS_CONFIG** | **LOCAL** | **LOCAL_A** | **LOCAL_B** | **LOCAL_DISPLAY** |
MODE | **OPERATE** | **OPTIONAL** | **OUTPUT** | **SERVICE** | **SPECIALIST** | **TEMPORARY** |
TUNE

Les attributs CLASS sont spécifiés dans le Tableau 16.

Tableau 16 – Attributs CLASS

Attribut	Description	Profils de communication
ALARM	VARIABLE contient des limites d'alarme.	CS, FF, GPE, HART, ISA100, PB, PN
ANALOG_INPUT	VARIABLE fait partie d'un bloc d'entrée analogique.	CS, GPE, PB, PN
ANALOG_OUTPUT	VARIABLE fait partie d'un bloc de sortie analogique.	CS, GPE, HART, PB, PN
COMPUTATION	VARIABLE fait partie d'un bloc de calcul.	CS, GPE, HART, PB, PN
CONSTANT	La valeur des attributs VARIABLE reste inchangée tout au long de la durée de vie de l'équipement. La valeur des attributs VARIABLE doit être conservée lorsqu'un équipement fait l'objet d'un redémarrage à chaud, d'un défaut d'alimentation ou même d'une réinitialisation de l'équipement aux paramètres d'usine. CLASS CONSTANT ne doit pas être combiné à DYNAMIC.	CS, FF, GPE, HART, ISA100, PB, PN
CONTAINED	VARIABLE représente les caractéristiques physiques de l'équipement. L'attribut VARIABLE représente un paramètre de bloc ne pouvant être référencé par une autre entrée de bloc ni défini par une sortie de bloc.	CS, FF, GPE, PB, PN
CORRECTION	VARIABLE fait partie du bloc de correction.	CS, GPE, HART, PB, PN
DEVICE	VARIABLE représente les caractéristiques physiques de l'équipement.	CS, GPE, HART, PB, PN
DIAGNOSTIC	VARIABLE indique l'état de l'équipement.	CS, FF, GPE, HART, ISA100, PB, PN
DIGITAL_INPUT	VARIABLE fait partie d'un bloc d'entrée numérique.	CS, GPE, PB, PN
DIGITAL_OUTPUT	VARIABLE fait partie d'un bloc de sortie numérique.	CS, GPE, PB, PN
DISCRETE	VARIABLE fait partie d'un bloc discret.	HART
DISCRETE_INPUT	VARIABLE fait partie d'un bloc d'entrée discret.	CS, GPE, PB, PN
DISCRETE_OUTPUT	VARIABLE fait partie d'un bloc de sortie discret.	CS, GPE, PB, PN

Attribut	Description	Profils de communication
DYNAMIC	L'attribut VARIABLE est modifié par l'équipement sans stimulus provenant du réseau. Les variables inscriptibles de CLASS DYNAMIC peuvent être modifiées. En cas de modification, la valeur affichée ne se met plus à jour et la modification est conservée dans la valeur affichée jusqu'à être appliquée sur l'équipement. Après application, la valeur de l'attribut VARIABLE recommence à se mettre à jour en continu. CLASS DYNAMIC ne doit pas être combiné à CONSTANT.	CS, FF, GPE, HART, ISA100, PB, PN
FREQUENCY	VARIABLE fait partie d'un bloc de fréquence.	HART
FREQUENCY_INPUT	VARIABLE fait partie d'un bloc d'entrée de fréquence.	CS, GPE, HART, PB, PN
FREQUENCY_OUTPUT	VARIABLE fait partie d'un bloc de sortie de fréquence.	CS, GPE, HART, PB, PN
HART	VARIABLE fait partie du bloc HART qui caractérise l'interface HART. Il y a un seul bloc HART.	CS, HART
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.	CS, FF, GPE, HART, ISA100, PB, PN
IS_CONFIG	Une modification de l'attribut VARIABLE entraîne la définition du compteur de révision ou du bit configuration-changed, selon le cas.	CS, FF, GPE, HART, PB, PN
LOCAL	Profils de communication: HART, FF, ISA100 Les attributs VARIABLE de CLASS LOCAL se comportent comme avec LOCAL_A. Profils de communication: CS, GPE, PB, PN Les attributs VARIABLE de CLASS LOCAL se comportent comme avec LOCAL_B.	CS, FF, GPE, HART, ISA100, PB, PN
LOCAL_A	L'attribut VARIABLE est utilisé localement par l'application EDD. Les attributs VARIABLE locaux ne sont pas stockés dans un équipement, mais ils peuvent être échangés avec un équipement. Les variables de CLASS LOCAL_A ne doivent pas appartenir à une autre classe, à l'exception de TEMPORARY. Les attributs VARIABLE de CLASS LOCAL_A sont automatiquement appliqués à l'ensemble de données actif après modification. L'ensemble de données actif correspond à l'ensemble de données en ligne ou hors ligne. Tous les caches fondés sur l'ensemble de données actif sont également synchronisés. Profils de communication: HART Lorsqu'un attribut VARIABLE de CLASS LOCAL_A est utilisé dans une COMMAND TRANSACTION, la variable doit inclure le qualificatif INFO.	CS, FF, GPE, HART, ISA100, PB, PN

Attribut	Description	Profils de communication
LOCAL_B	L'attribut VARIABLE est utilisé localement par l'application EDD. Les attributs VARIABLE locaux ne sont pas stockés dans un équipement, mais ils peuvent être échangés avec un équipement. Les variables de CLASS LOCAL_B ne doivent pas appartenir à une autre classe, à l'exception de TEMPORARY. Les attributs VARIABLE de CLASS LOCAL_B ne sont pas automatiquement appliqués à l'ensemble de données actif après modification. Ils sont appliqués simultanément aux variables de l'équipement. Profils de communication: HART Lorsqu'un attribut VARIABLE de CLASS LOCAL_B est utilisé dans une COMMAND TRANSACTION, la variable doit inclure le qualificatif INFO.	CS, FF, GPE, HART, ISA100, PB, PN
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'équipement.	CS, GPE, HART, PB, PN
MODE	VARIABLE contrôle le mode d'exploitation de l'équipement.	HART
OPERATE	L'attribut VARIABLE est utilisé pour contrôler l'exploitation d'un bloc (point de consigne, par exemple).	CS, FF, GPE, PB, PN
OPTIONAL	L'attribut VARIABLE peut ne pas être présent dans l'équipement. Un profil d'équipement définit les variables comme obligatoires ou facultatives. Si un équipement prend en charge une fonctionnalité, il doit la prendre en charge selon la définition de la variable facultative. Si un équipement ne prend pas en charge une variable facultative, l'application EDD ne doit pas afficher la valeur de ladite variable. Dans ce cas, l'application EDD peut masquer l'intégralité de la variable. Si une méthode lit une variable facultative non prise en charge, VARIABLE_STATUS est défini sur VARIABLE_STATUS_NOT_SUPPORTED et la valeur de la variable n'existe pas. Avant d'accéder à la valeur de la variable, la méthode doit vérifier l'attribut VARIABLE_STATUS sous peine d'échouer. Si une méthode écrit une variable facultative non prise en charge, VARIABLE_STATUS est défini sur VARIABLE_STATUS_NOT_SUPPORTED.	CS, GPE, PB, PN
OUTPUT	VARIABLE fait partie du bloc de sortie. Les valeurs d'attributs VARIABLE de sortie peuvent être accessibles par une autre entrée de bloc.	CS, FF, GPE, ISA100, PB, PN
SERVICE	L'attribut VARIABLE est utilisé lors de la conduite de la maintenance de routine.	CS, FF, GPE, HART, ISA100, PB, PN
SPECIALIST	VARIABLE ne doit être modifiable que si un spécialiste utilise l'application EDD. Pour un utilisateur non spécialiste, l'attribut VARIABLE avec cet attribut CLASS doit être en lecture seule indépendamment 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. Les attributs VARIABLE de CLASS SPECIALIST peuvent toujours être accessibles et modifiés à partir des METHODS.	CS, GPE, HART, ISA100, PB, PN

Attribut	Description	Profils de communication
TEMPORARY	L'attribut 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.	CS, GPE, PB, PN
TUNE	VARIABLE est utilisé pour régler l'algorithme d'un bloc.	CS, FF, GPE, HART, ISA100, PB, PN
Profils de communication: FF Une et une seule des classes: INPUT, OUTPUT, CONTAINED.		
Profils de communication: ISA100 CONSTANT et DYNAMIC peuvent être combinés avec zéro ou plusieurs: INPUT, OUTPUT, ALARM, DIAGNOSTIC, SERVICE, SPECIALIST, TUNE Lorsque ni DYNAMIC ni CONSTANT n'est défini, l'attribut VARIABLE nécessite uniquement d'être actualisé lorsqu'une modification est détectée.		
Profils de communication: HART En dehors de CLASS LOCAL ou CONSTANT, peut être combiné avec DYNAMIC, DIAGNOSTIC, SERVICE et IS_CONFIG. La valeur par défaut est sans CLASS. Dans ce cas, l'attribut VARIABLE n'est ni DYNAMIC, ni CONSTANT ni LOCAL; l'attribut VARIABLE est donc lu depuis l'équipement, mais nécessite uniquement d'être actualisé lorsqu'une modification est détectée.		

9.1.3 CONSTANT_UNIT

Objet

L'attribut CONSTANT_UNIT est utilisé si l'attribut VARIABLE a un code d'unité qui ne varie jamais (par exemple, l'amortissement est toujours en secondes). L'attribut CONSTANT_UNIT est spécifié comme une chaîne affichée avec la valeur de l'attribut VARIABLE. Un attribut VARIABLE sans unité constante ne comporte pas d'unité associée ou bien les unités ne sont pas constantes. Lorsque les unités techniques peuvent varier, elles doivent être représentées à l'aide d'une relation UNIT.

Structure lexicale

CONSTANT_UNIT *string-value* | *string-value*<cond>;

L'attribut CONSTANT_UNIT est spécifié dans le Tableau 17.

Tableau 17 – Attribut CONSTANT_UNIT

Type de valeurs	Description
string-value	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Chaîne de code d'unités à afficher.
string-value<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Condition qui spécifie la chaîne de code d'unités à afficher

9.1.4 DEFAULT_VALUE

Objet

L'attribut DEFAULT_VALUE spécifie le réglage par défaut de VARIABLE. L'attribut DEFAULT_VALUE permet à l'EDD de spécifier une valeur logique cohérente et raisonnable que l'application EDD peut utiliser lorsque l'équipement n'est pas physiquement en communication avec l'application EDD (par exemple, en cas de configuration d'un nouvel équipement hors ligne). Si aucun attribut INITIAL_VALUE n'est défini, l'attribut DEFAULT_VALUE est utilisé en tant qu'attribut INITIAL_VALUE.

Le type de l'expression doit être cohérent avec le type de VARIABLE. L'expression doit évaluer une constante sans exiger de communication avec l'équipement réel. L'attribut DEFAULT_VALUE peut également être spécifié par l'attribut TYPE de VARIABLE.

Pour les types arithmétiques, l'attribut DEFAULT_VALUE n'est pas à l'échelle.

Profils de communication: HART

Lorsque l'attribut DEFAULT_VALUE n'est pas spécifié, l'attribut VARIABLE doit prendre pour hypothèse la valeur répertoriée dans la colonne "Valeur initiale" du Tableau B.2 jusqu'à ce que l'utilisateur la modifie ou que la communication avec l'équipement modifie la valeur de l'attribut VARIABLE. Si la valeur initiale du Tableau B.2 n'est pas dans la plage MIN_VALUE à MAX_VALUE, elle est définie sur la valeur la plus proche de ladite plage.

Structure lexicale

Profils de communication: FF, ISA100, HART

DEFAULT_VALUE *expression*;

Profils de communication: CS, GPE, PB, PN

DEFAULT_VALUE *expression*<cond>;

Le type de l'expression doit être cohérent avec le type de VARIABLE.

9.1.5 INITIAL_VALUE

Objet

L'attribut INITIAL_VALUE spécifie la valeur d'un attribut VARIABLE qui est affiché lorsqu'un équipement 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.

La colonne "Valeur initiale" du Tableau B.2 présente les valeurs initiales en fonction du type de l'attribut VARIABLE.

Structure lexicale

INITIAL_VALUE *expression*;

Le type de l'expression doit être cohérent avec le type de VARIABLE.

9.1.6 POST_EDIT_ACTIONS

Objet

L'attribut POST_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés lorsque l'utilisateur a terminé de modifier l'attribut VARIABLE. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si la modification est annulée par l'utilisateur, l'attribut POST_EDIT_ACTIONS ne doit pas être exécuté. Si un attribut METHOD se termine pour une raison imprévue, les attributs METHOD suivants ne sont pas exécutés. Si l'un des attributs METHOD POST_EDIT_ACTIONS échoue, le cache de données est déjà mis à jour selon la nouvelle valeur et n'est pas affecté par l'échec.

Il convient d'utiliser l'attribut POST_EDIT_ACTIONS pour afficher des informations ou des avertissements, ou pour manipuler des variables. Les méthodes sont invoquées après modification de la valeur d'une variable.

Si un attribut METHOD POST_EDIT_ACTIONS échoue, il convient que l'application EDD avertisse l'utilisateur de l'anomalie.

Structure lexicale

```
POST_EDIT_ACTIONS { [actions-spezifier [, actions-spezifier]*] }
actions-spezifier:= [reference<cond> [, reference<cond>]*] |
                    [DEFINITION { compound-statement }]<cond>
```

Les attributs POST_EDIT_ACTIONS sont spécifiés dans le Tableau 18.

Tableau 18 – Attributs POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS

Attribut	Description
reference	Référence à une instance de METHOD ou à une instance de METHOD avec arguments.
DEFINITION	Profil de communication: CS, GPE, PB, PN Elément lexical (voir 14.4).

9.1.7 POST_READ_ACTIONS

Objet

L'attribut POST_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés après la lecture de l'attribut VARIABLE depuis l'équipement, à savoir après réception de la réponse de lecture, et doivent exploiter les données reçues. L'attribut POST_READ_ACTIONS est exécuté immédiatement après la réception des réponses et avant toute répartition des demandes de communication supplémentaires. 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 PRE_READ_ACTIONS échoue, l'attribut VARIABLE ne doit pas être lu depuis l'équipement et les attributs METHOD POST_READ_ACTIONS ne doivent pas être invoqués. Si l'un des attributs METHOD POST_READ_ACTIONS échoue, l'application EDD doit marquer la variable comme non lue.

Structure lexicale

```
POST_READ_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                    [DEFINITION { compound-statement }]<cond>
```

Les attributs POST_READ_ACTIONS sont spécifiés dans le Tableau 18.

9.1.8 POST_RQSTUPDATE_ACTIONS

Objet

POST_RQSTUPDATE_ACTIONS ne doit être pris en charge que par les simulateurs d'équipement. Lorsqu'il est présent, l'attribut POST_RQSTUPDATE_ACTIONS doit être ignoré dans toutes les applications EDD, à l'exception des simulateurs d'équipement.

L'attribut PREPOST_RQSTUPDATE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés quand la valeur de l'attribut VARIABLE est écrite par le biais d'une communication en provenance de l'application EDD. En d'autres termes, lorsque l'attribut VARIABLE est mis à jour par le biais d'un paquet de demandes (modifié ou non). Cela signifie que chaque action de mise à jour d'une variable après demande survient avant la mise à jour des variables qui la suivent dans le paquet de réception. Les actions doivent être exécutées et achevées avant la génération d'un paquet de réponses par le simulateur. 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<cond> [, reference<cond>]*
}
```

L'attribut POST_RQSTCHANGE_ACTIONS est spécifié dans le Tableau 19.

Tableau 19 – Attributs POST_USERCHANGE_ACTIONS et POST_RQSTUPDATE_ACTIONS

Attribut	Description
reference	Référence à un attribut METHOD.

9.1.9 POST_USERCHANGE_ACTIONS

Objet

POST_USERCHANGE_ACTIONS ne doit être pris en charge que par les simulateurs d'équipement. Lorsqu'il est présent, l'attribut POST_USERCHANGE_ACTIONS doit être ignoré dans toutes les applications EDD, à l'exception des simulateurs d'équipement.

L'attribut POST_USERCHANGE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés quand la modification de l'attribut VARIABLE dans l'application de simulation d'équipement est terminée. Cet attribut fonctionne comme l'attribut POST_EDIT_ACTIONS, sauf qu'il est uniquement exécuté si l'utilisateur modifie la valeur dans le simulateur d'équipement. 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<cond> [, reference<cond>]*
}
```

L'attribut POST_USERCHANGE_ACTIONS est spécifié dans le Tableau 19.

9.1.10 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'équipement, à savoir après réception de la réponse d'écriture, et doivent exploiter les données reçues, le cas échéant. L'attribut POST_WRITE_ACTIONS est exécuté immédiatement après la réception des réponses et avant toute répartition des demandes de communication supplémentaires. 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 PRE_WRITE_ACTIONS échoue, l'attribut VARIABLE ne doit pas être écrit sur l'équipement et les attributs METHOD POST_WRITE_ACTIONS ne doivent pas être invoqués. Si l'un des attributs METHOD POST_WRITE_ACTIONS échoue, l'application EDD doit marquer la variable comme non écrite.

Structure lexicale

```
POST_WRITE_ACTIONS { [actions-spezifier [, actions-spezifier]*] }
actions-spezifier:= [reference<cond> [, reference<cond>]*] |
    [DEFINITION { compound-statement }]<cond>
```

Les attributs POST_WRITE_ACTIONS sont spécifiés dans le Tableau 18.

9.1.11 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement lorsque l'attribut 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.

Il convient d'utiliser l'attribut PRE_EDIT_ACTIONS pour afficher des informations ou des avertissements adressés à l'utilisateur. Il convient de ne pas l'utiliser pour manipuler des variables. Les méthodes sont uniquement invoquées si l'utilisateur souhaite modifier la variable.

Si un attribut METHOD PRE_EDIT_ACTIONS échoue, il convient que l'application EDD avertisse l'utilisateur de l'anomalie et la modification doit être annulée.

Structure lexicale

```
PRE_EDIT_ACTIONS { [actions-spezifier [, actions-spezifier]*] }
actions-spezifier:= [reference<cond> [, reference<cond>]*] |
    [DEFINITION { compound-statement }]<cond>
```

Les attributs PRE_EDIT_ACTIONS sont spécifiés dans le Tableau 18.

9.1.12 PRE_READ_ACTIONS

Objet

L'attribut PRE_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant la lecture de VARIABLE. Les attributs PRE_READ_ACTIONS sont exécutés immédiatement avant la répartition des demandes de communication. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison inconnue, les attributs METHOD ultérieurs ne sont pas exécutés; l'opération de lecture ne doit pas être réalisée et échoue.

Structure lexicale

```
PRE_READ_ACTIONS { [actions-spezifier [, actions-spezifier]*] }
actions-spezifier:= [reference<cond> [, reference<cond>]*]|
                    [DEFINITION { compound-statement }]<cond>
```

Les attributs PRE_READ_ACTIONS sont spécifiés dans le Tableau 18.

9.1.13 PRE_WRITE_ACTIONS

Objet

L'attribut PRE_WRITE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant l'écriture de VARIABLE sur l'équipement. Les attributs PRE_WRITE_ACTIONS sont exécutés immédiatement avant la répartition des demandes de communication. Les attributs METHOD spécifiés doivent être exécutés dans l'ordre dans lequel ils apparaissent. Si un attribut METHOD se termine pour une raison inconnue, les attributs METHOD ultérieurs ne sont pas exécutés; l'opération d'écriture ne doit pas être réalisée et échoue.

Structure lexicale

```
PRE_WRITE_ACTIONS { [actions-spezifier [, actions-spezifier]*] }
actions-spezifier:= [reference<cond> [, reference<cond>]*]|
                    [DEFINITION { compound-statement }]<cond>
```

Les attributs PRE_WRITE_ACTIONS sont spécifiés dans le Tableau 18.

9.1.14 REFRESH_ACTIONS

Objet

Profils de communication: HART

Les attributs REFRESH_ACTIONS sont exécutés immédiatement avant l'affichage ou la mise à jour de l'attribut VARIABLE sur l'interface utilisateur. Les attributs REFRESH_ACTIONS sont exécutés au sein de méthodes (par exemple, lorsqu'un attribut VARIABLE est affiché dans PUT_MESSAGE, etc.). Si un attribut REFRESH_ACTION échoue, la valeur ne doit pas s'afficher.

Profils de communication: CS, GPE, PB, PN, FF, ISA100

L'attribut REFRESH_ACTIONS spécifie les attributs METHOD qui doivent être exécutés pour calculer la valeur de l'attribut VARIABLE sur la base d'autres attributs 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.

Les méthodes doivent être systématiquement invoquées avant qu'il soit nécessaire d'évaluer l'attribut VARIABLE (expressions conditionnelles, affichage, etc.).

L'attribut REFRESH_ACTIONS ne doit pas être utilisé pour modifier d'autres variables ou afficher un élément quelconque.

Structure lexicale

```
REFRESH_ACTIONS { [actions-spezifier [, actions-spezifier]*] }
actions-spezifier:= [reference<cond> [, reference<cond>]*]|
                    [DEFINITION { compound-statement }]<cond>
```

Les attributs REFRESH_ACTIONS sont spécifiés dans le Tableau 18.

9.1.15 TYPE

9.1.15.1 Structure générale

Objet

L'attribut TYPE décrit le format des valeurs de VARIABLE.

Structure lexicale

```
TYPE variable-type
variable-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 20.

Tableau 20 – Présentation des attributs TYPE

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.15.10	ASCII	✓	✓	✓	✓	✓	✓	✓
9.1.15.12	BITSTRING	✓	✓	✓	✗	✓	✓	✓
9.1.15.6	BIT_ENUMERATED	✓	✓	✓	✓	✓	✓	✓
9.1.15.13	BOOLEAN	✓	✓	✓	✗	✓	✓	✓
9.1.15.5	DATE	✓	✗	✓	✓	✗	✓	✓
9.1.15.5	DATE_AND_TIME	✓	✓	✓	✗	✗	✓	✓
9.1.15.4	DOUBLE	✓	✓	✓	✓	✓	✓	✓
9.1.15.5	DURATION	✓	✓	✓	✗	✓	✓	✓
9.1.15.7	ENUMERATED	✓	✓	✓	✓	✓	✓	✓
9.1.15.10	EUC	✓	✓	✓	✗	✓	✓	✓
9.1.15.4	FLOAT	✓	✓	✓	✓	✓	✓	✓
9.1.15.8	INDEX	✓	✓	✓	✓	✓	✓	✓
9.1.15.4	INTEGER	✓	✓	✓	✓	✓	✓	✓
9.1.15.9	OBJECT_REFERENCE	✓	✗	✓	✗	✗	✓	✓
9.1.15.11	OCTET	✓	✓	✓	✗	✓	✓	✓
9.1.15.10	PACKED_ASCII	✓	✗	✓	✓	✗	✓	✓
9.1.15.10	PASSWORD	✓	✓	✓	✓	✓	✓	✓
9.1.15.5	TIME	✓	✓	✓	✗	✓	✓	✓
9.1.15.5	TIME_VALUE(4)	✓	✗	✓	✓	✓	✓	✓
9.1.15.5	TIME_VALUE(8)	✓	✓	✓	✗	✗	✓	✓
9.1.15.4	UNSIGNED_INTEGER	✓	✓	✓	✓	✓	✓	✓
9.1.15.10	VISIBLE	✓	✓	✓	✗	✓	✓	✓

9.1.15.2 MIN_VALUE, MAX_VALUE

Objet

Les attributs MIN_VALUE et MAX_VALUE spécifient les limites inférieure et supérieure des valeurs sur lesquelles l'attribut VARIABLE doit être défini. Si la variable est dynamique (voir attribut CLASS), l'équipement peut définir la valeur de la variable hors de la plage spécifiée par ses attributs MIN_VALUE et MAX_VALUE. Les valeurs minimales et maximales sont comparées à la valeur non à l'échelle. La prise en charge des attributs MIN_VALUE et MAX_VALUE dépend de l'attribut TYPE. Les sous-sections de chaque type de variables ci-dessous identifient la structure lexicale et la prise en charge de chaque attribut TYPE.

Un attribut VARIABLE peut comporter plusieurs valeurs minimales/maximales; par exemple, pour spécifier plusieurs plages de validité (la variable peut par exemple avoir une plage juste au-dessus de zéro et une autre juste au-dessous, sans comprendre la valeur zéro). Lorsqu'il existe plusieurs valeurs minimales/maximales, un entier supplémentaire est utilisé pour numéroter chaque attribut MIN_VALUE/MAX_VALUE.

La plage spécifiée avec une paire d'attributs MIN_VALUE et MAX_VALUE peut ne pas chevaucher une autre plage spécifiée. 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 n'a pas besoin d'être spécifiée. L'attribut MIN_VALUE/MAX_VALUE peut également être construit par une expression (voir Article 18).

Profils de communication: FF, HART, ISA100

Les valeurs renvoyées par l'équipement doivent être acceptées et affichées, même lorsqu'elles dépassent les limites MIN_VALUE et MAX_VALUE.

Structure lexicale

[MAX_VALUE | MIN_VALUE] [n]? expression<cond>;

Le type de l'expression doit être cohérent avec le type de VARIABLE.

Les attributs MIN_VALUE et MAX_VALUE sont spécifiés dans le Tableau 21.

Tableau 21 – Attributs MIN_VALUE et MAX_VALUE

Type de valeurs	Description
n	Entier croissant constant de 1 à 9 qui identifie l'attribut MIN_VALUE ou MAX_VALUE lorsqu'il en existe plusieurs.

Utilisation

Les valeurs minimales et maximales d'une variable peuvent être référencées depuis un autre emplacement de l'EDD, défini en 19.2.

L'exemple suivant spécifie deux plages: la première de -10 à -5 et la seconde de 5 à 10.

```
MIN_VALUE1 -10;
MAX_VALUE1 -5;
MIN_VALUE2 5;
MAX_VALUE2 10;
```

9.1.15.3 value / description / help

Objet

L'attribut value comporte une constante qui spécifie la valeur. Le type dépend du type de données spécifié par l'attribut TYPE. Si l'attribut value est défini, l'attribut description est exigé. Des valeurs égales ne sont pas admises. La prise en charge des attributs value, description et help dépend de l'attribut TYPE. Les sous-sections de chaque type de variables ci-dessous identifient la structure lexicale et la prise en charge de chaque attribut TYPE.

L'attribut description comporte une brève description de l'élément.

L'attribut help spécifie le texte d'aide pour l'élément.

Structure lexicale

```
value:= [double-constant | float-constant | integer-constant | unsigned-integer-constant | string-constant]
description:= string-constant
help:= string-constant
```

9.1.15.4 Types arithmétiques

9.1.15.4.1 Structure générale

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, comme 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 double-value;
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
  INITIAL_VALUE double-constant;
  [MAX_VALUE[n]? double-value;]*
  [MIN_VALUE[n]? double-value;]*
  SCALING_FACTOR double-value;
  [{value, description, help}]<cond>]*
}]?
```

Structure lexicale – FLOAT

FLOAT

```
[{
  DEFAULT_VALUE float-value;
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
  INITIAL_VALUE float-constant;
  [MAX_VALUE[n]? float-value;]*
  [MIN_VALUE[n]? float-value;]*
  SCALING_FACTOR float-value;
  [{value, description, help}]<cond>]*
}]?
```


Structure lexicale – INTEGER

```

INTEGER (size)
[ {
    DEFAULT_VALUE integer-value;
    DISPLAY_FORMAT string-value;
    EDIT_FORMAT string-value;
    INITIAL_VALUE integer-constant;
    [MAX_VALUE[n]? integer-value;]*
    [MIN_VALUE[n]? integer-value;]*
    SCALING_FACTOR integer-value;
    [{value , description , help }]<cond>]*
}]?

```

Structure lexicale – UNSIGNED_INTEGER

```

UNSIGNED_INTEGER (size)
[ {
    DEFAULT_VALUE unsigned-integer-value;
    DISPLAY_FORMAT string-value;
    EDIT_FORMAT string-value;
    INITIAL_VALUE unsigned-integer-constant;
    [MAX_VALUE[n]? unsigned-integer-value;]*
    [MIN_VALUE[n]? unsigned-integer-value;]*
    SCALING_FACTOR unsigned-integer-value;
    [{value , description , help }]<cond>]*
}]?

```

Tableau 22 – Attributs TYPE pour DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.3	description	o	x	o	x	x	o	o
9.1.15.4.2	DISPLAY_FORMAT	o	o	o	o	o	o	o
9.1.15.4.3	EDIT_FORMAT	o	o	o	o	o	o	o
9.1.15.3	help	o	x	o	x	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.2	MAX_VALUE	o	o	o	o	o	o	o
9.1.15.2	MIN_VALUE	o	o	o	o	o	o	o
9.1.15.4.4	SCALING_FACTOR	o	o	o	o	o	o	o
9.1.15.4.5	size	o	o	o	o	o	o	o
9.1.15.3	value	o	x	o	x	x	o	o

9.1.15.4.2 DISPLAY_FORMAT**Objet**

Spécifie comment la valeur de l'attribut VARIABLE est affichée. La chaîne DISPLAY_FORMAT doit être définie conformément à la fonction printf du langage de programmation C sans le qualificatif-préfixe "%" (voir l'ISO/IEC 9899:1999). Lorsqu'un attribut DISPLAY_FORMAT n'est pas spécifié, le type arithmétique de la variable détermine la chaîne de format par défaut décrite dans le Tableau 23.

Tableau 23 – Type de conversion du format d'affichage par défaut

Type arithmétique	Format d'affichage
INTEGER	d
UNSIGNED_INTEGER	u
FLOAT	g
DOUBLE	g

Profils de communication: CS, GPE, PB, PN

Lorsque le spécificateur de format par défaut du Tableau 23 est utilisé, la largeur et la précision par défaut sont uniquement limitées par le type arithmétique de l'attribut VARIABLE.

Profils de communication: HART, FF, ISA100

Lorsque le spécificateur de format par défaut du Tableau 23 est utilisé, la largeur et la précision par défaut décrites dans le Tableau 24 sont également utilisées.

Tableau 24 – Largeur/précision par défaut fondées sur l'attribut VARIABLE et le type de conversion de format

Type	Type de conversion	Taille ^a	Largeur	Précision
FLOAT	f, F, e, E, g, G	n/a	12	5
	d, l, o, u	n/a	11	0
	x	n/a	8	0
DOUBLE	f, F, e, E, g, G	n/a	18	8
	d, l, o, u	n/a	15	0
	x	n/a	16	0
INTEGER UNSIGNED_INTEGER	f, F, e, E, g, G	n/a	12	6
		d, l, u	1	4
		2	6	0
		4	11	0
		8	20	0
	o	1	3	0
		2	6	0
		4	11	0
		8	22	0
	x	1	2	0
		2	4	0
		4	8	0
		8	16	0
	^a n/a = non applicable			

Structure lexicale

DISPLAY_FORMAT *string-value*<cond>;

9.1.15.4.3 EDIT_FORMAT

Objet

Spécifie comment la valeur de l'attribut VARIABLE est modifiée. La chaîne EDIT_FORMAT doit être définie conformément à la fonction printf du langage de programmation C (voir l'ISO/IEC 9899:1999). Un format de modification ne comporte pas de qualificatif-préfixe "%" et contrôle uniquement les éléments balayés lors de l'analyse d'une valeur modifiée par l'utilisateur. Lorsqu'un attribut EDIT_FORMAT n'est pas spécifié, la chaîne DISPLAY_FORMAT utilisée pour la variable doit être utilisée.

Structure lexicale

EDIT_FORMAT *string-value*<cond>;

Le format de modification est constitué d'un spécificateur de largeur numérique facultatif, d'un point décimal facultatif et d'un spécificateur de précision suivi d'un spécificateur de type.

flags width* [.precision]* type*

Les fanions peuvent comporter plusieurs spécificateurs, y compris les spécificateurs dièse (#) et zéro (0). Leur utilisation est identique à celle de la fonction ANSI C printf.

Le spécificateur de largeur est un entier décimal qui définit le nombre maximal de caractères à entrer ou balayer. Lorsque la largeur n'est pas spécifiée, la valeur par défaut admet la valeur positive la plus élevée et la valeur négative la plus faible que la variable prend en compte comme valeur entrée par l'utilisateur. Ce spécificateur est précisément le même que le spécificateur de largeur utilisé dans la fonction ANSI C scanf.

Le spécificateur de précision facultatif est un entier décimal qui définit le nombre maximal de chiffres utilisables à droite du point décimal. Ce spécificateur est ignoré lorsque le spécificateur de type se trouve dans l'entrée d'entier "diouxX" définie et doit être inférieur au spécificateur de largeur (le cas échéant). Lorsque le spécificateur de précision est absent de la chaîne du format de modification, aucune restriction d'entrée de données (en dehors du spécificateur de largeur, le cas échéant) n'est appliquée aux chiffres entrés à droite du point décimal.

Le spécificateur de type est l'un des types d'entrées "diouxXeEfFgG" spécifiés pour la fonction scanf.

9.1.15.4.4 SCALING_FACTOR

Objet

L'attribut SCALING_FACTOR indique que la valeur affichée de l'attribut VARIABLE n'est pas la valeur renvoyée par l'équipement. La valeur affichée est la valeur retournée par un équipement multipliée par un facteur. Par conséquent, le processeur EDDL doit multiplier la valeur de l'attribut VARIABLE renvoyée par l'équipement par son SCALING_FACTOR avant d'être affichée. L'attribut SCALING_FACTOR ne doit pas être égal à zéro.

L'attribut SCALING_FACTOR peut également être généré par une expression (voir Article 18). L'attribut SCALING_FACTOR est uniquement appliqué à des fins d'affichage. A tout autre moment (par exemple, lors de l'accès à l'attribut VARIABLE depuis une méthode ou de sa comparaison à une valeur minimale/maximale), la valeur reste équivalente à (et au même format que) la valeur non à l'échelle renvoyée par l'équipement. Les valeurs minimales et maximales utilisées avec le facteur d'échelle sont également des valeurs non à l'échelle.

L'attribut SCALING_FACTOR ne doit pas affecter la chaîne DISPLAY_FORMAT d'un attribut VARIABLE.

Structure lexicale

SCALING_FACTOR [*double-value*<cond> | *double-expression*<cond>];

9.1.15.4.5 size

Objet

Taille du type de données en octets. La taille est une constante entière de 1 à 8 inclus. Cette valeur est facultative. La valeur par défaut est 1.

9.1.15.5 DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

9.1.15.5.1 Structure générale

Objet

Les attributs VARIABLE de type de données DATE, DATE_AND_TIME, DURATION, TIME ou 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 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 millisecondes (ms) et d'un compteur de jours;
- le type de données TIME est constitué d'une heure et d'une date;
- 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

Le type de données DATE spécifie que l'attribut VARIABLE est un entier non signé de trois octets (Day, Month, Year). Le codage des octets est spécifié en B.2.2.

DATE

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

Pour l'attribut DEFAULT_VALUE, l'expression constante doit être résolue par un entier non signé de 3 octets où le MSByte est Day et le LSByte est (Year-1900). Par exemple, le 24 juin 1993 peut s'exprimer comme suit:

```
DEFAULT_VALUE (( (24<<8)+6)<<8)+93;
```

Structure lexicale – DATE_AND_TIME

L'attribut DATE_AND_TIME spécifie que l'attribut VARIABLE est un entier non signé de sept octets représentant la date calendaire (date et heure). Le codage des octets est spécifié en B.2.3.

DATE_AND_TIME

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

Structure lexicale – DURATION

L'attribut DURATION spécifie que l'attribut VARIABLE est un entier non signé de 6 octets. Les quatre premiers bits doivent avoir la valeur zéro. Les bits 0 à 27 représentent l'heure en millisecondes (ms) à partir de minuit. Le nombre de jours est une valeur binaire de 16 bits. Le codage des octets est spécifié en B.2.4.

DURATION

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

Structure lexicale – TIME

L'attribut TIME spécifie que l'attribut VARIABLE est un entier non signé de six octets. L'heure est exprimée en millisecondes (ms) à partir de minuit. A minuit, le compteur recommence à zéro.

La date est indiquée en jours par rapport au 1984-01-01. Le premier jour de janvier 1984, le compteur commence à la valeur zéro. Le codage des octets est spécifié en B.2.5.

TIME

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
}]?
```

Structure lexicale – TIME_VALUE

L'attribut TIME_VALUE spécifie que l'attribut VARIABLE est un entier non signé de quatre ou huit octets. Ce nombre à décimale fixe représente le temps exprimé en multiples de 1/32 ms. Le codage des octets est spécifié en B.2.6.

TIME_VALUE (size)

```
[{
  DEFAULT_VALUE unsigned-integer-value;
  INITIAL_VALUE unsigned-integer-constant;
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
  [MAX_VALUE[n]? unsigned-integer-value;]*
  [MIN_VALUE[n]? unsigned-integer-value;]*
  TIME_FORMAT string-value;
  TIME_SCALE time-scale-specifier;
}]?
```

Pour l'attribut DEFAULT_VALUE, l'expression constante doit être résolue par un entier non signé de 4 octets ou de 8 octets. Par exemple, un codage TIME_VALUE de 4 octets pour 5 h 14 min 26 s peut s'exprimer comme suit:

```
DEFAULT_VALUE ((5*60+14)*60+26)*32000;
```

Les attributs MIN_VALUE et MAX_VALUE doivent être résolus par un entier non signé de 4 octets ou 8 octets (comme représenté pour l'attribut DEFAULT_VALUE) ou par une référence à un autre attribut VARIABLE de TYPE TIME_VALUE.

Les attributs des types DATE, DATE_AND_TIME, DURATION, TIME et TIME_VALUE sont spécifiés dans le Tableau 25.

Tableau 25 – Attributs TYPE pour DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	m	x	m	x	x	m	m
9.1.15.4.2	DISPLAY_FORMAT	o	x	o	o	x	o	o
9.1.15.4.3	EDIT_FORMAT	o	x	o	o	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.2	MAX_VALUE	o	o	o	o	o	o	o
9.1.15.2	MIN_VALUE	o	o	o	o	o	o	o
9.1.15.5.4	size	o	x	o	x	x	o	o
9.1.15.5.2	TIME_FORMAT	o	o	o	o	o	o	o
9.1.15.5.3	TIME_SCALE	o	x	o	o	x	o	o

9.1.15.5.2 TIME_FORMAT

Objet

L'attribut TIME_FORMAT spécifie comment l'attribut VARIABLE est affiché et modifié. 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 %d %D %e %F %g %G %h %m %x %y %Y.

Lorsque TIME_FORMAT est spécifié, DISPLAY_FORMAT et EDIT_FORMAT ne doivent pas être spécifiés.

Structure lexicale

TIME_FORMAT *string-value*<cond>

9.1.15.5.3 TIME_SCALE

Objet

L'attribut TIME_SCALE spécifie comment l'attribut VARIABLE est mis à l'échelle, ce qui peut correspondre à SECONDS, MINUTES ou HOURS. 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. Sinon, seul l'attribut TIME_FORMAT peut être présent.

Si l'attribut TIME_SCALE est absent, l'attribut TIME_FORMAT par défaut doit être "%T"; sinon, le format doit être par défaut le même que celui de l'attribut VARIABLE de TYPE FLOAT.

L'attribut TIME_SCALE est un facteur d'échelle; les règles de l'attribut SCALING_FACTOR sont applicables.

Structure lexicale

TIME_SCALE [SECONDS , MINUTES , HOURS]<cond>

9.1.15.5.4 size

Objet

Taille du type de données en octets. Doit être de 4 ou 8. Pour FF, la valeur par défaut est 8 et pour les autres protocoles, la valeur est 4. Cet attribut est applicable uniquement à TIME_VALUE.

9.1.15.6 BIT_ENUMERATED

9.1.15.6.1 Structure générale

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.

Un bit individuel dans un attribut VARIABLE de type BIT_ENUMERATED peut être référencé depuis un autre emplacement de l'EDD par le nom de la variable et le masque de bits spécifié en 16.3.

A l'instar de la valeur de la déclaration, un masque de bits doit être une constante entière spécifiant un bit unique.

Structure lexicale

```
BIT_ENUMERATED (size)
{
    DEFAULT_VALUE unsigned-integer-value;
    INITIAL_VALUE unsigned-integer-constant;
    [[{ value , description , help , class , status-class , action } ,]<cond>]*
    [{ value , description , help , class , status-class , action }]<cond>
}
```

Les attributs de type BIT_ENUMERATED sont spécifiés dans le Tableau 26.

Tableau 26 – Attributs TYPE pour BIT_ENUMERATED

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.15.6.4	action	o	o	o	o	o	o	o
9.1.2	class	o	o	o	o	o	o	o
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.6.2	description	m	m	m	m	m	m	m
14.8	help	o	o	o	o	o	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.6.5	size	o	o	o	o	o	o	o
9.1.15.6.6	status-class	o	o	o	o	o	o	o
9.1.15.6.3	value	m	m	m	m	m	m	m

9.1.15.6.2 description

Objet

La description est la chaîne affichée pour ce bit.

9.1.15.6.3 value

Objet

Entier non signé qui représente la signification du bit. Si la valeur de l'attribut VARIABLE affecte aux positions de bit spécifiées un un (1) logique, la description associée est affichée. Des valeurs égales ne sont pas admises.

9.1.15.6.4 action

Objet

Référence à un attribut METHOD qui doit être exécuté quand le bit est défini. L'attribut METHOD doit être exécuté lorsque le bit est défini (passe de 0 à 1) au cours d'une opération de lecture ou lorsque le bit est lu pour la première fois comme un 1. Lorsque le bit est défini et que l'action est exécutée, l'action n'est pas renouvelée jusqu'à ce que le bit repasse à 0, puis à 1.

9.1.15.6.5 size

Objet

Taille du type de données en octets. La taille est une constante entière de 1 à 8 inclus. Cette valeur est facultative. La valeur par défaut est 1.

Tableau 27 – Attribut size

Type de valeurs	Description
unsigned-integer	Profils de communication: CS, GPE, HART, PB, PN Constante entière non signée supérieure à zéro.
unsigned-integer	Profils de communication: FF, ISA100 Constante entière non signée supérieure à zéro (limitée à 4 au maximum).

9.1.15.6.6 status-class

Objet

Signification du bit. Zéro ou plusieurs classes d'états peuvent être définies pour un bit (voir Tableau 28). Les classes d'états sont organisées en catégories; une seule classe d'états de chaque catégorie peut être définie pour un bit. Pour un bit donné, zéro ou une valeur peut être assignée à partir de chaque catégorie. Lorsque plusieurs classes d'états sont présentes, elles sont séparées par le symbole "&". Si l'attribut VARIABLE n'est pas un octet d'état, les bits n'ont pas de classe d'état.

Le Tableau 28 présente les attributs status-class pour un attribut VARIABLE de type BIT_ENUMERATED.

Tableau 28 – Attributs status–class

Catégorie	Attribut	Description
Cause:		
Les classes d'états de cause indiquent à une application EDD les raisons de la définition du bit. Zéro ou plusieurs classes d'états peuvent être appliquées à un bit individuel.	HARDWARE	Défaillance du matériel.
	SOFTWARE	Défaillance du logiciel.
	PROCESS	Problème avec le processus connecté à l'équipement.
	MODE	L'équipement est dans un mode particulier.
	DATA	Configuration de données non valide.
	MISC	Conditions diverses.
Durée:		
Les classes d'états de durée indiquent à une application EDD la durée pendant laquelle le bit peut être défini. Zéro ou une classe d'état peut être appliquée à un bit individuel.	EVENT	Événement unique.
	STATE	L'équipement est dans un état particulier.
Corrigibilité:		
Les classes d'états de corrigibilité indiquent à une application EDD si le problème signalé par le bit défini peut être corrigé. Zéro ou une classe d'état peut être appliquée à un bit individuel.	SELF_CORRECTING	Le bit est réinitialisé sans intervention supplémentaire.
	CORRECTABLE	Le bit peut être effacé par l'application EDD ou le processus connecté.
	UNCORRECTABLE	Le bit n'est ni auto-correctible, ni corrigible.
Domaine d'application:		
Les classes d'états de domaine d'application indiquent si bit récapitule un problème dans l'équipement ou fournit des informations supplémentaires pour clarifier le problème. Zéro ou une classe d'état peut être appliquée à un bit individuel.	SUMMARY	Le bit est une combinaison logique d'autres bits de classe DETAIL.
	DETAIL	Le bit est répété ailleurs dans un bit de classe SUMMARY.
Divers:		
Ces classes d'états incluent celles qui n'entrent pas dans l'une des autres classes d'états. Zéro ou plusieurs classes d'états peuvent être appliquées à un bit individuel.	MORE	Des informations d'état supplémentaires sont disponibles en provenance de l'équipement.
	COMM_ERROR	Défaillance de la communication dans l'autre équipement.
	IGNORE_IN_TEMPORARY_MASTER	Il convient d'ignorer un bit dans les équipements maîtres temporaires.

Catégorie	Attribut	Description
Sévérité:		
Les classes d'états de sévérité indiquent l'importance ou l'urgence qu'une application EDD doit appliquer au bit lorsqu'il est défini par l'équipement. Un seul des états INFO, WARNING ou ERROR doit être appliqué à un bit individuel. Leur combinaison avec l'état IGNORE_IN_HOST ou IGNORE_IN_HANDHELD peut être associée à un bit.	INFO	Le bit donne des informations sur l'état du processus ou de l'équipement.
	WARNING	Le bit donne un avertissement sur l'état du processus ou de l'équipement.
	ERROR	Le bit donne une erreur sur l'état du processus ou de l'équipement.
	IGNORE_IN_HOST	Le bit doit être ignoré dans une application EDD fixe, raccordée en permanence.
	IGNORE_IN_HANDHELD	Il convient d'ignorer un bit soit ignoré dans les appareils de communication portatifs.
Etat de sortie:		
	ALL	Affecte toutes les sorties (voir Tableau 29).
	AO	Affecte une des sorties analogiques (voir Tableau 29).
	BAD	La sortie n'est pas fiable, il convient de ne pas l'utiliser pour la commande (voir Tableau 29).
	DV	Affecte une des variables dynamiques (voir Tableau 29).
	TV	Affecte une des variables d'émetteur (voir Tableau 29).

Les attributs status-class ALL, AO, DV et TV donnent des informations d'état supplémentaires sur la sortie d'un équipement.

Structure lexicale – ALL

ALL [AUTO_BAD, AUTO_GOOD, MANUAL_BAD, MANUAL_GOOD]

Structure lexicale – AO

AOinteger ([AUTO & BAD | AUTO & GOOD | MANUAL & BAD | MANUAL & GOOD])

Structure lexicale – DV

DVinteger ([AUTO & BAD | AUTO & GOOD | MANUAL & BAD | MANUAL & GOOD])

Structure lexicale – TV

TVinteger ([AUTO & BAD | AUTO & GOOD | MANUAL & BAD | MANUAL & GOOD])

Les attributs sont spécifiés dans le Tableau 29.

Tableau 29 – Attributs ALL, AO, DV, TV

Valeur d'attribut	Description
AUTO & BAD	La valeur provient d'un processus d'application et n'est pas valide.
AUTO & GOOD	La valeur provient d'un processus d'application et est valide.
integer	La valeur de VARIABLE qui provient de l'équipement.
MANUAL & BAD	La valeur ne provient pas d'un processus d'application et n'est pas valide.
MANUAL & GOOD	La valeur ne provient pas d'un processus d'application et est valide.

Utilisation

Une classe de sortie est une combinaison des valeurs AUTO ou MANUAL et GOOD, MARGINAL ou BAD. Par conséquent, les classes de sorties valides sont:

AUTO & GOOD	AUTO & MARGINAL	AUTO & BAD
MANUAL & GOOD	MANUAL & MARGINAL	MANUAL & BAD

Si une sortie est AUTO, elle provient du processus connecté; si une sortie est MANUAL, elle ne provient pas du processus connecté et est sous le contrôle d'un être humain. Si une sortie est GOOD, elle est fiable et peut être utilisée à des fins de contrôle; si une sortie est BAD, elle n'est pas fiable et il convient de ne pas l'utiliser à des fins de contrôle.

Si le bit d'état est

DV1 (AUTO & BAD)

la variable primaire provient du processus et n'est pas fiable.

Si le bit d'état est

AO1 (MANUAL & GOOD)

la première sortie analogique ne provient pas du processus et est fiable.

Si le bit d'état est

TV6 (MANUAL & BAD)

la sixième variable de l'équipement ne provient pas du processus et n'est pas fiable.

Si le bit d'état est

ALL (AUTO & BAD)

toutes les sorties de l'équipement proviennent du processus et ne sont pas fiables.

9.1.15.7 ENUMERATED

9.1.15.7.1 Structure générale

Objet

Un attribut VARIABLE de type ENUMERATED est un entier non signé défini comme un ensemble de valeurs avec chaînes de texte associées.

Structure lexicale

```
ENUMERATED (size)
{
    DEFAULT_VALUE unsigned-integer-value;
    INITIAL_VALUE unsigned-integer-constant;
    [{ value , description , help } , ]<cond>*
    [{ value , description , help }]<cond>
}
```

Les attributs de type ENUMERATED sont spécifiés dans le Tableau 30.

Tableau 30 – Attributs TYPE pour ENUMERATED

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.3	description	m	m	m	m	m	m	m
9.1.15.3	help	o	o	o	o	o	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.7.2	size	o	o	o	o	o	o	o
9.1.15.3	value	m	m	m	m	m	m	m

9.1.15.7.2 size**Objet**

Taille du type de données en octets. La taille est une constante entière de 1 à 8 inclus. Cette valeur est facultative. La valeur par défaut est 1.

9.1.15.8 INDEX**Objet**

Un attribut VARIABLE de TYPE INDEX est un entier non signé qui est utilisé pour référencer les éléments de VALUE_ARRAY, de REFERENCE_ARRAY (voir 9.1) et de LIST (voir 9.1). Un attribut VARIABLE de TYPE INDEX peut uniquement prendre les valeurs définies par l'attribut REFERENCE_ARRAY, VALUE_ARRAY ou LIST.

Lorsque l'index est lié à un attribut LIST, la variable d'index doit automatiquement avoir VALIDITY FALSE lorsque la liste est vide.

Lorsqu'un attribut VARIABLE INDEX est lié à un attribut VALUE_ARRAY ou LIST, et présenté à l'utilisateur, la valeur numérique doit être affichée.

Lorsqu'un attribut VARIABLE INDEX est lié à un attribut REFERENCE_ARRAY et présenté à l'utilisateur, le texte associé aux indices de la matrice doit être affiché, mais pas la valeur numérique de l'attribut VARIABLE.

Structure lexicale

INDEX [(size)]? [reference | reference<cond>;

Les attributs de type index sont spécifiés dans le Tableau 31.

Tableau 31 – Attributs TYPE pour INDEX

Type de valeurs	Description
size	Taille de l'attribut VARIABLE, en octets. Taille du type de données en octets. La taille est une constante entière de 1 à 4 inclus. Cette valeur est facultative. La valeur par défaut est 1. L'attribut REFERENCE_ARRAY ou LIST doit contenir uniquement des éléments qui ne dépassent pas la taille d'index de la variable. Profils de communication: HART Un attribut INDEX de taille 1 doit uniquement représenter des valeurs de 0 à 249. La valeur de 250 est largement utilisée avec HART pour indiquer "non utilisé" dans les commandes indexées comme les commandes 33 et 105. L'attribut INDEX de taille 2 n'a pas de restrictions similaires.
reference	Référence à une instance de l'attribut REFERENCE_ARRAY, VALUE_ARRAY ou LIST.
reference<cond>	Profils de communication: CS, GPE, PB, PN Référence à une instance de l'attribut REFERENCE_ARRAY ou LIST.

9.1.15.9 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 équipement modulaire (voir l'IEC 61804-4).

Structure lexicale**OBJECT_REFERENCE****9.1.15.10 ASCII, EUC, PACKED_ASCII, PASSWORD, VISIBLE****9.1.15.10.1 Structure générale****Objet**

Les variables de chaîne comprennent les types suivants:

- ASCII – est utilisé pour spécifier une séquence de caractères du jeu de caractères ISO Latin-1;
- 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). EUC doit utiliser le format de codage UTF-16, limité aux seuls caractères du plan multilingue de base (BMP, plan 0);
- PACKED_ASCII – est un sous-ensemble d'ASCII obtenu en enlevant les deux premiers bits de poids fort de chaque caractère ASCII (voir B.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 mots de passe et ASCII sont identiques;
- VISIBLE – Cette chaîne est une séquence ordonnée de caractères du jeu de caractères numéro 2 de l'ISO 646, plus le caractère ESPACE. Elle est limitée à des valeurs de 7 bits dans un octet de 8 bits (le bit de poids fort est défini sur zéro).

Structure lexicale – EUC

```

EUC (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
  } ]?

```

Structure lexicale – ASCII

```

ASCII (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
  } ]?

```

Structure lexicale – PACKED_ASCII

```

PACKED_ASCII (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
  } ]?

```

Structure lexicale – PASSWORD

```

PASSWORD (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
  } ]?

```

Structure lexicale – VISIBLE

```

VISIBLE (size)
[ {
    DEFAULT_VALUE string-value;
    INITIAL_VALUE string-constant;
    [{value , description , help }]<cond>*
  } ]?

```

Les attributs des types de chaînes sont spécifiés dans le Tableau 32.

Tableau 32 – Attributs TYPE pour ASCII, EUC, PACKED_ASCII, PASSWORD, VISIBLE

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.3	description	o	x	o	x	x	o	o
9.1.15.3	help	o	x	o	x	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.10.2	size	m	m	m	m	m	m	m
9.1.15.3	value	o	x	o	x	x	o	o

L'application EDD doit afficher les types de données ASCII, EUC, PACKED_ASCII, PASSWORD et VISIBLE comme des chaînes. La terminaison ou le rognage des types de données ne modifie pas la taille de l'élément de données. La terminaison et le rognage sont utilisés pour la gestion des chaînes uniquement dans l'application EDD. Même si le type de données peut être terminé, les caractères restants de l'élément de données sont remplis par les caractères de remplissages correspondants. En fonction du type de données, la terminaison, les caractères de remplissage et les caractères de fin doivent être traités comme le montre le Tableau 33.

Tableau 33 – Gestion de chaîne

TYPE	Terminaison	Caractère de remplissage	Caractères de fin (rognage)
ASCII	Null	Null	Non
EUC	Null	Null	Non
PACKED_ASCII	Non terminé	Espace	Non
PASSWORD	Profils de communication: HART, PB, PN: Null Profils de communication: FF, ISA100: non terminé	Profils de communication: HART, PB, PN: Null Profils de communication: FF, ISA100: espace	Non
VISIBLE	Non terminé	Espace	Oui

9.1.15.10.2 size

Objet

Pour le type de chaîne EUC, size spécifie le nombre de paires d'octets (mots de 16 bits) réservé à la représentation en mémoire de la variable. Le codage UTF-16 étant utilisé et les caractères limités aux caractères du plan BMP 0, le nombre de caractères pouvant être représentés dans la chaîne EUC est identique à la valeur de l'attribut size.

En dehors du type de chaîne EUC, size spécifie le nombre de caractères du jeu de caractères applicable, lequel peut être différent de la taille de la représentation en mémoire de la variable.

La taille est une constante entière supérieure à zéro. Cette valeur n'est pas facultative; une taille doit être spécifiée.

9.1.15.11 OCTET

9.1.15.11.1 Structure générale

Objet

L'attribut OCTET est utilisé pour spécifier une séquence de données binaires non formatées dont la définition est déterminée par la mise en œuvre de l'équipement.

Structure lexicale – OCTET

OCTET (*size*)

```
[{
  DEFAULT_VALUE [{ integer-constant [, integer-constant]* } |
                  string-value ]<cond>;
  INITIAL_VALUE [{ integer-constant [, integer-constant]* } |
                  string-constant ];
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
}]?
```

Les attributs DEFAULT_VALUE et INITIAL_VALUE peuvent prendre l'une de ces deux formes:

- si l'attribut OCTET est destiné à être utilisé comme une séquence d'octets de données non formatées, les *integer-constants* doivent être résolues par une valeur de 0 à 255;
- si l'attribut OCTET est destiné à être utilisé comme une séquence de caractères du jeu de caractères ISO Latin-1 ou UTF-8, la *string-value* ou la *string-constant* est utilisée.

Les attributs de type OCTET sont spécifiés dans le Tableau 34.

Tableau 34 – Attributs TYPE pour OCTET

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.15.11.2	DISPLAY_FORMAT	o	o	o	x	o	o	o
9.1.15.11.3	EDIT_FORMAT	o	o	o	x	o	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.11.4	size	m	m	m	x	m	m	m

9.1.15.11.2 DISPLAY_FORMAT

Objet

Lorsqu'il est spécifié, l'attribut DISPLAY_FORMAT spécifie comment la valeur de l'attribut VARIABLE est affichée.

Profils de communication: CS, GPE, PB, PN

Lorsque l'attribut DISPLAY_FORMAT n'est pas spécifié, le format par défaut "#x" doit être utilisé.

Profils de communication: ISA100, FF

Lorsque l'attribut DISPLAY_FORMAT n'est pas spécifié, le format par défaut "X" doit être utilisé.

L'attribut `DISPLAY_FORMAT` "s" force l'interprétation de l'attribut `VARIABLE` de type OCTET comme une séquence de caractères du jeu de caractères ISO Latin-1 et son affichage comme une chaîne.

L'attribut `DISPLAY_FORMAT` "x" ou "X" force l'interprétation de l'attribut `VARIABLE` de type OCTET comme une séquence d'octets de données non formatées et son affichage comme une chaîne de valeurs au format hexadécimal (par exemple, "12 ab cd", etc.). L'utilisation de la minuscule "x" doit entraîner l'affichage des valeurs et préfixes en lettres minuscules. Sinon, l'utilisation de la majuscule "X" doit entraîner l'affichage en lettres majuscules. Le fanion de début # doit entraîner l'affichage du préfixe hexadécimal (#x pour "0x", #X pour "0X").

L'attribut `DISPLAY_FORMAT` "o" force l'interprétation de l'attribut `VARIABLE` de type OCTET comme une séquence d'octets de données non formatées et son affichage comme une chaîne de valeurs au format d'entier non signé (par exemple, "12 50 67", etc.). Le fanion de début # doit entraîner l'affichage du préfixe octal "0".

L'attribut `DISPLAY_FORMAT` "d" ou "i" force l'interprétation de l'attribut `VARIABLE` de type OCTET comme une séquence d'octets de données non formatées et son affichage comme une chaîne de valeurs au format d'entier signé (par exemple, "-10 123 -40", etc.).

L'attribut `DISPLAY_FORMAT` "u" force l'interprétation de l'attribut `VARIABLE` de type OCTET comme une séquence d'octets de données non formatées et son affichage comme une chaîne de valeurs au format d'entier non signé (par exemple, "10 123 40", etc.).

L'attribut `DISPLAY_FORMAT` "U" force l'interprétation de l'attribut `VARIABLE` de type OCTET comme une séquence de caractères du jeu de caractères UTF-8 (RFC 3629). L'utilisation du format "U" intervient en plus du format d'affichage utilisé selon l'ISO/IEC 9899.

Structure lexicale

`DISPLAY_FORMAT` *string-value*<cond>

9.1.15.11.3 EDIT_FORMAT

Objet

Spécifie le format de modification. Lorsqu'un attribut `EDIT_FORMAT` n'est pas spécifié, la chaîne `DISPLAY_FORMAT` utilisée pour la variable doit être utilisée. Les formats de modification admis sont identiques à ceux définis pour l'attribut `DISPLAY_FORMAT` en 9.1.15.11.2.

Structure lexicale

`EDIT_FORMAT` *string-value*<cond>;

9.1.15.11.4 size

Objet

La taille spécifie le nombre d'octets.

La taille est une constante entière supérieure à zéro. Cette valeur n'est pas facultative; une taille doit être spécifiée.

9.1.15.12 BITSTRING

9.1.15.12.1 Structure générale

Objet

L'attribut BITSTRING est utilisé pour spécifier une séquence de bits. Lorsqu'elles sont présentées à des fins d'affichage ou de modification, les valeurs BITSTRING peuvent être représentées comme une série de bits (1's et 0's) ou comme une séquence d'octets représentés au format hexadécimal.

Structure lexicale – BITSTRING

BITSTRING (*size*)

```
[{
  DEFAULT_VALUE { integer-constant [, integer-constant]* }<cond>;
  INITIAL_VALUE { integer-constant [, integer-constant]* };
  DISPLAY_FORMAT string-value;
  EDIT_FORMAT string-value;
}]?
```

Pour les attributs DEFAULT_VALUE et INITIAL_VALUE, les *integer-constants* doivent être résolues par une valeur comprise entre 0 et 255, chaque valeur correspondant à un octet.

Les attributs de type BITSTRING sont spécifiés dans le Tableau 35.

Tableau 35 – Attributs TYPE pour BITSTRING

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
9.1.4	DEFAULT_VALUE	o	x	o	x	x	o	o
9.1.5	INITIAL_VALUE	o	x	o	x	x	o	o
9.1.15.12.2	size	m	m	m	x	m	m	m

9.1.15.12.2 size

Objet

La taille spécifie le nombre de bits, laquelle doit être un multiple de 8.

La taille est une constante entière supérieure à zéro. Cette valeur n'est pas facultative; une taille doit être spécifiée.

9.1.15.13 BOOLEAN

Objet

Un attribut VARIABLE de type BOOLEAN est un entier non signé d'un octet avec la valeur TRUE ou FALSE. La valeur 0 correspond à FALSE. Se reporter au Tableau B.12 pour connaître la valeur correspondant à TRUE.

Structure lexicale

BOOLEAN

9.2 REFERENCE_ARRAY

9.2.1 Structure générale

Objet

Un attribut REFERENCE_ARRAY est un ensemble d'éléments du type d'élément spécifié (par exemple, VARIABLE ou MENU). Chaque élément du groupe a un numéro assigné, appelé index.

L'attribut REFERENCE_ARRAY a deux mots-clés disponibles, "ITEM_ARRAY" (première option) et "ARRAY" (deuxième option), qui peuvent être choisis par un profil. Le mot-clé ITEM_ARRAY ne doit pas être choisi si le paramètre VALUE_ARRAY est utilisé.

Structure lexicale

Profils de communication: CS, GPE, FF, ISA100, PB, PN

reference-array-keyword := **ITEM_ARRAY**

Profils de communication: CS, GPE, HART, PB, PN

reference-array-keyword := **ARRAY**

```
reference-array-keyword OF item-type identifier
{
  LABEL string-value;
  HELP string-value;
  ELEMENTS { [[integer , reference [, description [, help]? ]?;]<cond>]+ }
  PRIVATE boolean-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}
```

Les attributs REFERENCE_ARRAY sont spécifiés dans le Tableau 36.

Tableau 36 – Attributs REFERENCE_ARRAY

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		REFERENCE_ARRAY	✓	✓	✓	✓	✓	✓	✓
9.2.2		ELEMENTS	m	m	m	m	m	m	m
14.8		HELP	o	o	o	o	o	o	o
9.2.3		item-type	m	m	m	m	m	m	m
14.9		LABEL	o	o	o	o	o	o	o
14.13		PRIVATE	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o

Utilisation

L'exemple de la Figure 30 représente la définition de l'attribut REFERENCE_ARRAY intitulé deviceVariables.

```

VARIABLE pressureValue
{
    CLASS CORRECTION & DYNAMIC;
    TYPE FLOAT;
}

VARIABLE pressureUnits
{
    TYPE ENUMERATED (2)
    {
        ALL_UNITS
    }
}

VARIABLE temperatureValue
{
    CLASS CORRECTION & DYNAMIC;
    TYPE FLOAT;
}

VARIABLE temperatureUnits
{
    TYPE ENUMERATED (2)
    {
        ALL_UNITS
    }
}

COLLECTION OF VARIABLE pressure
{
    MEMBERS
    {
        DIGITAL_VALUE,          pressureValue;
        DIGITAL_UNITS,          pressureUnits;
    }
}

COLLECTION OF VARIABLE temperature
{
    MEMBERS
    {
        DIGITAL_VALUE,          temperatureValue;
        DIGITAL_UNITS,          temperatureUnits;
    }
}

ARRAY OF COLLECTION deviceVariables
{
    ELEMENTS
    {
        0, pressure;
        1, temperature;
    }
}

```

Figure 30 – Exemple d'attribut REFERENCE_ARRAY

L'exemple de la Figure 31 représente la construction d'une définition de commande abstraite.

```

COMMAND read_dynamic_variables_and_pv_current
{
    REDEFINE TRANSACTION
    {
        REQUEST { }
        REPLY
        {
            response_code, device status,
            dynamic_variables[0].DAQ.ANALOG_VALUE,
            dynamic_variables[0].DEVICE_VARIABLE.DIGITAL_UNITS,
            dynamic_variables[0].DEVICE_VARIABLE.DIGITAL_VALUE
        }
    }
}

```

Figure 31 – Exemple de définition de commande abstraite

Dans cet exemple, `dynamic_variables[0].DEVICE_VARIABLE.DIGITAL_VALUE` fait référence à `pressureValue`.

9.2.2 ELEMENTS

Objet

L'attribut **ELEMENTS** spécifie une liste d'éléments. Il définit pour chaque élément la référence d'élément, son index associé, une description et une aide facultatives. Pour une description de la concaténation des aides et des libellés, se reporter à l'IEC 61804-4.

Structure lexicale

ELEMENTS { [[*integer* , *reference* [, *description* [, *help*] ?]? ;] <cond>]+ }

Les attributs **ELEMENTS** sont spécifiés dans le Tableau 37.

Tableau 37 – Attribut ELEMENTS

Type de valeurs	Description
integer	Numéro par lequel l'élément peut être référencé. Chaque numéro doit être défini de manière unique dans une construction <code>REFERENCE_ARRAY</code> .
reference	Référence à un élément EDD. Le Tableau 46 présente les attributs item-types admis.
description	Brève description pour l'élément.
help	Texte d'aide pour l'élément.

9.2.3 item-type

Objet

L'attribut **item-type** spécifie le type de membres dans l'attribut `REFERENCE_ARRAY`. Tous les membres de l'attribut `REFERENCE_ARRAY` doivent être du type spécifié.

Structure lexicale

item-type

Le Tableau 46 présente les attributs item-types admis.

9.3 VALUE_ARRAY

9.3.1 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 par l'identificateur de **VALUE_ARRAY** et l'index de l'élément.

Structure lexicale

ARRAY *identifier*

```
{
    LABEL string-value;
    HELP string-value;
    NUMBER_OF_ELEMENTS [integer-constant | unsigned-integer-constant |
                        integer-expression];
    PRIVATE boolean-specifier;
    RESPONSE_CODES reference;
    TYPE reference;
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
    WRITE_MODE write-mode-specifier;
}
```

Les attributs VALUE_ARRAY sont spécifiés dans le Tableau 38.

Tableau 38 – Attributs VALUE_ARRAY

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		VALUE_ARRAY	✓	✓	✓	✓	✓	✓	✓
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	m	m	m	m	m	m	m
9.3.2		NUMBER_OF_ELEMENTS	m	m	m	m	m	m	m
14.13		PRIVATE	o	o	o	o	o	o	o
14.15		RESPONSE_CODES	o	o	o	x	o	o	o
9.3.3		TYPE	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o
14.20		WRITE_MODE	x	o	x	x	o	x	x

9.3.2 NUMBER_OF_ELEMENTS

Objet

L'attribut NUMBER_OF_ELEMENTS spécifie le nombre d'éléments dans l'attribut VALUE_ARRAY en tant qu'entier ou qu'entier non signé supérieur à zéro.

Structure lexicale

```
NUMBER_OF_ELEMENTS [integer-constant | unsigned-integer-constant |
                    integer-expression];
```

Les attributs NUMBER_OF_ELEMENTS sont spécifiés dans le Tableau 39.

Tableau 39 – Attribut NUMBER_OF_ELEMENTS

Type de valeurs	Description
integer-constant	Profils de communication: FF, ISA100 Nombre d'éléments d'un attribut VALUE_ARRAY.
unsigned-integer-constant	Profils de communication: HART Nombre d'éléments d'un attribut VALUE_ARRAY.
integer-expression	Profils de communication: CS, GPE, PB, PN Expression qui doit être évaluée à une constante entière non négative qui se situe dans la plage d'index de l'attribut NUMBER_OF_ELEMENTS (pour la description des expressions, voir Article 18).

9.3.3 TYPE

Objet

L'attribut TYPE est une référence à une instance de construction de base EDD. 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 40.

Tableau 40 – Attribut TYPE

Attribut	Description
reference	Référence à une instance de construction de base 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. Profils de communication: FF, ISA100 Référence à un attribut VARIABLE. Profils de communication: CS, GPE, HART, PB, PN Référence à un attribut VARIABLE, COLLECTION, LIST, VALUE_ARRAY ou REFERENCE_ARRAY.

Utilisation

Prendre, par exemple, ce qui suit:

```
VARIABLE xValue { TYPE FLOAT; }
VARIABLE yValue { TYPE FLOAT; }
VARIABLE quality {
    TYPE enumerated { {0 , "Bad"}, {2 , "Poor"}, {3 , "Good"} }
}

COLLECTION point { MEMBERS { X, xValue; Y, yValue; QUALITY, quality; }

ARRAY dataSetArray { NUMBER_OF_ELEMENTS 100; TYPE point; }
```

Cela permet de créer 100 ensembles de 3 variables. En d'autres termes, 300 variables sont créées dans la matrice.

9.4 LIST

9.4.1 Structure générale

Objet

L'attribut LIST est une matrice de valeurs insérables de longueur variable. L'attribut LIST décrit une séquence continue d'éléments dans laquelle chaque élément de la séquence est défini par une instance de construction EDDL de type spécifique.

Structure lexicale

```
LIST identifier
{
  LABEL string-value;
  HELP string-value;
  CAPACITY [unsigned-integer-constant |
            unsigned-integer-expression<cond>];
  COUNT [unsigned-integer-constant | unsigned-integer-constant<cond> |
         unsigned-integer-expression<cond> | reference];
  PRIVATE boolean-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  TYPE reference;
}
```

Les attributs LIST sont spécifiés dans le Tableau 41.

Tableau 41 – Attributs LIST

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	LIST		✓	✓	✓	✓	✓	✓	✓
9.4.2		CAPACITY	o	o	o	o	o	o	o
9.4.3		COUNT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.13		PRIVATE	o	o	o	o	o	o	o
9.4.4		TYPE	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o

Utilisation

Un élément de liste est référencé depuis un autre emplacement dans l'EDD par l'identificateur de la liste et l'index de l'élément. La notation de cette référence utilise des crochets. En raison de la nature dynamique des listes, une référence de liste ne doit pas être utilisée dans une expression. La syntaxe de cette référence est décrite en 16.10.

Les éléments d'une liste sont toujours contigus et toutes les références de liste doivent être correctes et bien formées. L'accès à un élément LIST inexistant dans un attribut METHOD doit entraîner l'échec de la méthode. Les références à des éléments LIST inexistants à un autre emplacement de l'EDD doivent se comporter comme si l'élément référencé avait VALIDITY FALSE. Les propriétés de liste peuvent également être référencées, y compris COUNT, CAPACITY, FIRST et LAST.


```
list-identifieur.COUNT
list-identifieur.CAPACITY
list-identifieur.FIRST
list-identifieur.LAST
```

La propriété COUNT renvoie le nombre d'éléments qui résident actuellement dans les listes. La propriété CAPACITY indique le nombre d'éléments pouvant être stockés dans la liste.

Profils de communication: CS, GPE, HART, PB, PN

L'utilisation de la propriété FIRST équivaut à référencer list-identifieur [0] et celle de la propriété LAST à référencer list-identifieur[list-identifieur.COUNT – 1].

Profils de communication: FF, ISA100

L'utilisation de la propriété FIRST équivaut à référencer list-identifieur [1] et celle de la propriété LAST à référencer list-identifieur[list-identifieur.COUNT].

Les listes sont généralement gérées à l'aide de méthodes et des builtins correspondants spécifiés dans l'IEC 61804-5. Ces builtins renvoient des codes d'erreur en cas de tentative d'accès à un élément de liste inexistant.

9.4.2 CAPACITY

Objet

L'attribut CAPACITY spécifie le nombre maximal d'éléments qui peuvent être stockés dans la liste. Lorsque l'attribut CAPACITY n'est pas spécifié, il est par défaut défini sur la valeur unsigned-integer maximale (+4 294 967 295), qui indique que l'attribut LIST n'est pas limité.

Structure lexicale

CAPACITY [*unsigned-integer-constant* | *unsigned-integer-expression*<cond>];

L'attribut CAPACITY est spécifié dans le Tableau 42.

Tableau 42 – Attribut CAPACITY

Type de valeurs	Description
unsigned-integer-constant	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Constante entière non signée supérieure ou égale à un, et inférieure ou égale à la valeur entière non signée maximale (+4 294 967 295)
unsigned-integer-expression<cond>	Profils de communication: FF, ISA100 Expression d'entier non signé; peut-être définie par une condition

9.4.3 COUNT

Objet

L'attribut COUNT spécifie le nombre d'éléments actuellement stockés dans la liste. Lorsqu'un attribut 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 la propriété COUNT est spécifiée comme une constante, elle définit la taille lors de la création de l'instance de LIST. Si elle n'est pas définie ou qu'elle est définie comme une référence, une instance de LIST avec zéro élément est créée.

Structure lexicale

COUNT [*unsigned-integer-constant* | *unsigned-integer-constant*<cond> | *unsigned-integer-expression*<cond> | *reference*];

L'attribut COUNT est spécifié dans le Tableau 43.

Tableau 43 – Attribut COUNT

Type de valeurs	Description
unsigned-integer-constant	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Constante entière non signée
unsigned-integer-constant<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Constante entière non signée; peut-être définie par une condition
unsigned-integer-expression<cond>	Profils de communication: FF, ISA100 Expression d'entier non signé; peut-être définie par une condition
reference	Référence à une instance de l'attribut VARIABLE

L'attribut COUNT, lorsqu'il fait référence à une variable non-LOCAL, force l'équipement à contrôler la taille de la liste (la liste est gérée par l'équipement).

Les builtins ListInsert() et ListDeleteElementAt() forcent la mise à jour de la variable référencée par l'attribut COUNT afin de correspondre au nombre réel d'éléments dans l'attribut LIST.

Si l'attribut COUNT est résolu par une expression, aucune des variables référencées dans l'expression n'est mise à jour du fait des builtins.

Il ne doit pas être admis d'écrire l'attribut COUNT dans un attribut METHOD.

Chaque fois que la variable référencée par l'attribut COUNT varie, la synchronisation de l'intégralité de l'attribut LIST est exigée, car la structure LIST n'est plus valide.

Pour lire un attribut LIST géré par l'équipement, l'application EDD doit relire tous les éléments de l'attribut LIST depuis l'équipement chaque fois que la valeur de la variable référencée par l'attribut COUNT lue depuis l'équipement varie.

Pour écrire un attribut LIST géré par l'équipement sur l'équipement, l'application EDD doit écrire la variable référencée par l'attribut COUNT et tous les éléments de l'attribut LIST lorsque la variable référencée par l'attribut COUNT varie dans le cache (par un attribut METHOD, par exemple). Lorsque le contenu de l'attribut LIST a été modifié, mais qu'aucun élément n'a été ajouté ou supprimé, seuls les éléments de l'attribut LIST modifié nécessitent d'être écrits sur l'équipement.

9.4.4 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 LIST.

Structure lexicale

TYPE *reference*;

L'attribut TYPE est spécifié dans le Tableau 44.

Tableau 44 – Attribut TYPE

Type de valeurs	Description
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. La référence doit se terminer sur un élément de données. L'élément référencé 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.

9.5 COLLECTION

9.5.1 Structure générale

Objet

Un attribut COLLECTION est un groupe logique d'éléments, tels que des attributs VARIABLE ou MENU. Un identificateur est assigné à chaque élément. Les éléments de l'attribut COLLECTION peuvent être référencés depuis un autre emplacement dans l'EDD par l'identificateur de COLLECTION et le nom de MEMBER en utilisant la notation par point:

collection-identifiant.member-name

Chaque nom d'élément utilisé dans un attribut COLLECTION doit être unique par rapport à tous les autres noms d'élément définis dans le même attribut COLLECTION.

L'attribut COLLECTION est destiné à être utilisé pour identifier des paramètres qu'il convient de traiter conjointement, par exemple.

Structure lexicale

```

COLLECTION [OF item-type]? identifiant
{
  LABEL string-value;
  HELP string-value;
  MEMBERS { members-specifier }
  PRIVATE boolean-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}

```

Les attributs COLLECTION sont spécifiés dans le Tableau 45.

Tableau 45 – Attributs COLLECTION

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		COLLECTION	✓	✓	✓	✓	✓	✓	✓
14.8		HELP	o	o	o	o	o	o	o
9.5.2		item-type	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
14.13		PRIVATE	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o

9.5.2 item-type

Objet

L'attribut item-type spécifie le type de membres dans la collection. Si item-type est spécifié, tous les membres de la collection doivent être du type spécifié. Sinon, les membres de la collection peuvent être d'un type quelconque.

Profils de communication: HART

Utilisation déconseillée; prise en charge pour la rétrocompatibilité

Structure lexicale

item-type

Le Tableau 46 présente les attributs item-types admis.

Tableau 46 – item-type

item-type	Description
BLOCK_A	Profils de communication: FF, ISA100 Sélection de cette construction de base.
BLOCK_B	Profils de communication: CS, GPE, PB, PN Sélection de cette construction de base.
CHART	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
COLLECTION	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
EDIT_DISPLAY	Profils de communication: FF, HART, ISA100 Sélection de cette construction de base.
FILE	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
GRAPH	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.

item-type	Description
GRID	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
IMAGE	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
LIST	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
MENU	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
METHOD	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
PLUGIN	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
RECORD	Profils de communication: FF, ISA100 Sélection de cette construction de base.
REFERENCE_ARRAY	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
VALUE_ARRAY	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
VARIABLE	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Sélection de cette construction de base.
VARIABLE_LIST	Profils de communication: FF, ISA100 Sélection de cette construction de base.

9.6 FILE

9.6.1 Structure générale

Objet

FILE décrit une zone de stockage persistant qui est gérée par l'application EDD. L'attribut FILE doit uniquement contenir 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. Lorsque l'attribut IDENTITY n'est pas présent, l'attribut FILE est associé à une instance d'un équipement.

Un attribut MEMBER peut être une référence à un attribut VARIABLE, LIST ou VALUE_ARRAY. L'attribut FILE peut uniquement contenir des données; aucune de ces données ne peut être de classe DYNAMIC.

NOTE Si l'attribut MEMBER n'est pas de CLASS LOCAL, les données de l'attribut FILE sont écrasées s'il est connecté en ligne à l'équipement.

Structure lexicale

```

FILE identifier
{
    LABEL string-value;
    HELP string-value;
    HANDLING handling-specifier;
    IDENTITY file-name;
    MEMBERS { members-specifier }
    ON_UPDATE_ACTIONS { [reference<cond>]+ }
    SHARED [TRUE | FALSE];
}

```

Les attributs FILE sont spécifiés dans le Tableau 47.

Tableau 47 – Attributs FILE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	FILE		✓	✓	✓	✓	✓	✓	✓
14.6		HANDLING	o	x	o	o	x	o	o
14.8		HELP	o	o	o	o	o	o	o
9.6.2		IDENTITY	o	x	o	o	x	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
9.6.3		ON_UPDATE_ACTIONS	x	x	x	o	x	x	x
9.6.4		SHARED	o	x	o	o	x	o	o

9.6.2 IDENTITY

Objet

L'attribut IDENTITY désigne le fichier informatique ou l'espace de stockage pour un attribut FILE partagé donné et est utilisé par les applications EDD pour l'identifier. Cela signifie également que l'ensemble de données externe peut être consulté par toutes les instances de type d'équipement en conséquence. Il incombe à l'application EDD d'accéder à l'attribut FILE approprié lorsque l'application EDD accède à un attribut IDENTITY. L'attribut IDENTITY est une chaîne de caractères ASCII entre guillemets constituée de lettres (a-z et A-Z), de chiffres (0-9), du caractère de soulignement (_) ou d'un point (.). Afin de s'assurer que l'attribut IDENTITY de l'attribut FILE ne soit pas confondu avec celui éventuellement utilisé par d'autres équipements, les applications EDD doivent séparer leur espace de stockage pour les attributs FILE par type d'équipement et, en fonction de la valeur de l'équipement SHARED, éventuellement par instance d'équipement.

Si l'attribut SHARED est spécifié et que l'attribut IDENTITY n'est pas spécifié, l'identité de l'attribut FILE peut être établie à l'aide de la fonction builtin browselidentity(). L'attribut IDENTITY est directement accessible depuis l'EDD à l'aide de la chaîne de référence

file_reference.IDENTITY

(qui renvoie une chaîne de caractères correspondant à l'attribut IDENTITY de l'attribut FILE).

Structure lexicale

```

IDENTITY file-name;

```

L'attribut IDENTITY est spécifié dans le Tableau 48.

Tableau 48 – Attribut IDENTITY

Type de valeurs	Description
file-name	Littéral de chaîne qui identifie le nom de fichier. Profils de communication: HART Longueur limitée à 32 caractères

9.6.3 ON_UPDATE_ACTIONS**Objet**

ON_UPDATE_ACTIONS ne doit être pris en charge que par les simulateurs d'équipement.

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*<cond>]+ }

L'attribut ON_UPDATE_ACTIONS est spécifié dans le Tableau 49.

Tableau 49 – Attribut ON_UPDATE_ACTIONS

Type de valeurs	Description
reference	Référence à un attribut METHOD.

9.6.4 SHARED**Objet**

L'attribut SHARED spécifie si le fichier est partagé par toutes les instances du type d'équipement. Si SHARED n'est pas spécifié ou si sa valeur est FALSE, le fichier est associé à l'instance d'équipement spécifique.

Les fichiers partagés sont généralement des référentiels de données d'ingénierie précalculées (par exemple, géométries de réservoir ou tableaux de flux).

NOTE Pour les profils de communication: HART: Les fichiers partagés sont fondés sur XML et conformes au schéma XML contrôlé et fourni par le FieldComm Group. Contacter le FieldComm Group pour plus d'informations.

Structure lexicale

SHARED [TRUE | FALSE];

Les attributs SHARED sont spécifiés dans le Tableau 50.

Tableau 50 – Attribut SHARED

Valeur d'attribut	Description
TRUE	Le fichier est partagé par toutes les instances du même type d'équipement.
FALSE	Le fichier est spécifique à l'instance d'équipement. La valeur par défaut de SHARED est FALSE.

9.7 Relations

9.7.1 REFRESH

Objet

L'écriture d'un paramètre sur un équipement peut provoquer la mise à jour par l'équipement 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, et envoyée à l'équipement. La modification peut être initiée par des interactions de l'utilisateur dans l'application EDD, dans les exécutions de METHOD ou par la lecture d'une nouvelle valeur depuis l'équipement. Afin d'empêcher l'exécution infinie de relations d'actualisation, les affected-parameters doivent uniquement être lus si le cause-parameter a une valeur différente de celle qu'il avait lors de sa dernière lecture. Au minimum, l'actualisation doit être réalisée avant d'utiliser ou d'afficher l'une des variables dépendant de la relation d'actualisation.

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 l'IEC 61804-4).

Profils de communication: FF, ISA100, CS, GPE, PB, PN

Un attribut VARIABLE peut avoir une relation REFRESH avec lui-même, ce qui signifie que VARIABLE doit être lu après écriture. Il convient de ne pas placer de valeur dynamique dans une relation avec elle-même, car cela peut entraîner des résultats imprévus.

Structure lexicale

```

REFRESH identifier
{ [cause-parameter] | [cause-parameter<cond> [, cause-parameter]<cond>]+] :
  [affected-parameter [, affected-parameter]+] |
  [affected-parameter<cond> [, affected-parameter<cond>]+]
}
  
```

La relation REFRESH est spécifiée dans le Tableau 51.

Tableau 51 – Présentation de la relation REFRESH

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	REFRESH		✓	✓	✓	✓	✓	✓	✓

Les attributs REFRESH sont spécifiés dans le Tableau 52.

Tableau 52 – Attributs REFRESH

Type de valeurs	Description
cause-parameter	Profils de communication: FF, ISA100 Instance de l'attribut VARIABLE, RECORD ou VALUE_ARRAY. Profils de communication: HART Instance de l'attribut VARIABLE, LIST ou VALUE_ARRAY.
cause-parameter<cond> [, cause-parameter]<cond>]+	Profils de communication: CS, GPE, PB, PN Ensemble d'instances de l'attribut VARIABLE ou VALUE_ARRAY. Peut-être défini par une condition.
affected-parameter	Profils de communication: FF, ISA100 Ensemble d'instances de l'attribut VARIABLE, LIST, RECORD ou VALUE_ARRAY qui doivent être lues, 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. Profils de communication: HART Ensemble d'instances de l'attribut VARIABLE, LIST ou VALUE_ARRAY qui doivent être lues, 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.
affected-parameter<cond>	Profils de communication: CS, GPE, PB, PN Ensemble d'instances de l'attribut VARIABLE, LIST, RECORD ou VALUE_ARRAY qui doivent être lues, si un des paramètres de cause a été modifié. Peut-être défini par une condition. Un attribut VARIABLE de ce jeu de paramètres affectés peut également être dans le jeu de paramètres de cause d'une autre relation REFRESH.

9.7.2 SEMANTIC_MAP

Objet

Les relations SEMANTIC_MAP fournissent un accès générique aux éléments de données par le biais des identificateurs sémantiques normalisés, sans connaître aucun nom d'élément EDD interne. Les identificateurs sémantiques sont des chaînes qui sont indépendantes des paramètres régionaux et sont généralement spécifiées dans les Normes internationales comme le Dictionnaire de données commun (CDD) de l'IEC (IEC 61987). Afin d'éviter tout conflit avec différents espaces de noms dans ces identificateurs, la structure de la chaîne utilisée doit être fondée sur l'ISO/TS 29002-5.

Un exemple d'identificateur sémantique est "0112/2///61987#ABB271#002", l'identificateur sémantique de la propriété "nom d'étiquette/nom de demande PCE".

Une relation SEMANTIC_MAP définit une paire "clé/valeur", où la "clé" comporte l'identificateur sémantique et la "valeur" la référence à l'élément de données lié.

Si l'élément de données référencé est une VARIABLE d'énumération, la référence à l'élément de données peut être étendue avec une liste d'énumération pour le mapping des valeurs d'énumération vers les identificateurs sémantiques.

Si l'unité de l'élément de données référencé est spécifiée comme un attribut `CONSTANT_UNIT`, la référence à l'élément de données peut être étendue en ajoutant l'attribut `CONSTANT_UNIT` et l'identificateur sémantique correspondant à l'unité.

Structure lexicale

SEMANTIC_MAP *identifier*

```
{
  [semantic-id] | [semantic-id [, semantic-id]+]:
  [semantic-item [, semantic-item]+]
}
semantic-item:= reference [{ [{ value , semantic-id-enum } , ]*
                           [{ value , semantic-id-enum }]]?
                           [CONSTANT_UNIT semantic-id-unit;]? }]?

```

La relation `SEMANTIC_MAP` est spécifiée dans le Tableau 53.

Tableau 53 – Présentation de la relation SEMANTIC_MAP

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	SEMANTIC_MAP		✓	✓	✓	✓	✓	✓	✓

Les attributs `SEMANTIC_MAP` sont spécifiés dans le Tableau 54.

Tableau 54 – Attributs SEMANTIC_MAP

Type de valeurs	Description
semantic-id	Littéral de chaîne spécifié comme un identificateur sémantique
semantic-item	Référence à l'élément de données lié
semantic-id-enum	Littéral de chaîne spécifié comme un identificateur sémantique pour une valeur d'énumération
semantic-id-unit	Littéral de chaîne spécifié comme un identificateur sémantique pour une unité

Utilisation

La Figure 32 représente plusieurs définitions `SEMANTIC_MAP`.

```
// --- Exemple 1 ---
SEMANTIC_MAP map_tag
{
    "0112/2///61987#ABB271#002":
    var_tag
}

// --- Exemple 2 ---
SEMANTIC_MAP map_damping
{
    "<semID_damp_A>", "<semID_damp_B>", "<semID_damp_C>":
    var_damping
    {
        CONSTANT_UNIT "<semID_seconds>";
    }
}

// --- Exemple 3 ---
SEMANTIC_MAP map_meas_type
{
    "<semID_meas_type>":
    var_meas_type
    {
        { 2, "<semID_meas_type_absPress>" }, // Type de mesure: pression différentielle
        { 3, "<semID_meas_type_diffPress>" } // Type de mesure: pression absolue
    }
}
```

Figure 32 – Exemples d'EDD d'attributs SEMANTIC_MAP

Exemple 1:

représente une relation de un à un pour le mapping de l'identificateur sémantique "0112/2///61987#ABB271#002", l'identificateur sémantique de la propriété "nom d'étiquette/nom de demande PCE", vers la VARIABLE qui gère l'étiquette de l'équipement.

Exemple 2:

représente une relation de n à 1 pour le mapping de plusieurs identificateurs sémantiques (3 identificateurs dans cet exemple) vers une VARIABLE. Etant donné que l'unité de l'attribut VARIABLE "var_damping" est spécifiée comme un attribut CONSTANT_UNIT, la référence à "var_damping" a été étendue avec l'attribut CONSTANT_UNIT et l'identificateur spécifique pour l'unité "secondes".

Exemple 3:

représente une relation pour le mapping d'un identificateur sémantique vers une VARIABLE d'énumération. En outre, les valeurs des listes d'énumération sont associées par mapping aux identificateurs sémantiques correspondants.

9.7.3 UNIT

Objet

Une relation UNIT spécifie une référence à un attribut VARIABLE contenant une unité et une liste d'éléments dépendants. Lorsqu'un élément affecté est affiché, son unité doit également être affichée.

Structure lexicale

```
UNIT identifier
{ [cause-parameter | cause-parameter<cond>]:
  [affected-parameter [, affected-parameter]+] |
  [affected-parameter<cond> [, affected-parameter<cond>]+] }
}
```

La relation UNIT est spécifiée dans le Tableau 55.

Tableau 55 – Présentation de la relation UNIT

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	UNIT		✓	✓	✓	✓	✓	✓	✓

Les attributs UNIT sont spécifiés dans le Tableau 56.

Tableau 56 – Attributs UNIT

Type de valeurs	Description
cause-parameter	Profils de communication: FF, HART, ISA100 Référence à une instance de VARIABLE de type énumérée contenant le code d'unité ou à une instance de VARIABLE de type chaîne contenant la chaîne d'unités.
cause-parameter<cond>	Profils de communication: CS, GPE, PB, PN Référence à une instance de VARIABLE de type énumérée contenant le code d'unité ou à une instance de VARIABLE de type chaîne contenant la chaîne d'unités. Peut-être défini par une condition.
affected-parameter	Profils de communication: FF, ISA100 Liste d'instances de l'attribut VARIABLE, VALUE_ARRAY, LIST ou AXIS qui utilisent ce code d'unité. Profils de communication: HART Liste d'instances de l'attribut VARIABLE ou AXIS qui utilisent ce code d'unité.
affected-parameter<cond>	Profils de communication: CS, GPE, PB, PN Liste d'instances de l'attribut VARIABLE, VALUE_ARRAY, LIST ou AXIS qui utilisent ce code d'unité. Peut-être défini par une condition.

9.7.4 WRITE_AS_ONE

Objet

Une relation WRITE_AS_ONE spécifie une liste d'éléments de données qui doivent être modifiés ensemble. L'application EDD doit afficher tous les éléments de la relation lorsque l'un de ces éléments est modifié, 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 équipement exige que des éléments de données spécifiques soient analysés et modifiés simultanément pour une exploitation correcte. Si les variables peuvent être modifiées séparément, il convient de ne pas utiliser de relation WRITE_AS_ONE.

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 identifier
{
  [reference [, reference]]+ | [reference<cond> [, reference<cond>]]+
}
```

La relation WRITE_AS_ONE est spécifiée dans le Tableau 57.

Tableau 57 – Présentation de la relation WRITE_AS_ONE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	WRITE_AS_ONE		✓	✓	✓	✓	✓	✓	✓

L'attribut WRITE_AS_ONE est spécifié dans le Tableau 58.

Tableau 58 – Attribut WRITE_AS_ONE

Attribut	Description
reference	Profils de communication: FF, ISA100 Liste des instances des attributs VARIABLE, VALUE_ARRAY et RECORD qui doivent être affichées. Profils de communication: HART La liste des attributs VARIABLE qui ne sont pas BIT_ENUMERATED doit être affichée. Les éléments VALUE_ARRAY et LIST de TYPE VARIABLE qui ne sont pas BIT_ENUMERATED doivent être affichés.
reference<cond>	Profils de communication: CS, GPE, PB, PN Liste des instances des attributs VARIABLE et VALUE_ARRAY qui doivent être affichées.

9.8 BLOB

9.8.1 Structure générale

Objet

Le BLOB (Binary Large Object) doit être utilisé pour transférer des données binaires depuis ou vers un équipement. L'application EDD a un accès en lecture seule au contenu d'un attribut BLOB, uniquement à des fins de validation, par exemple. Un BLOB est un conteneur de données dont le contenu est accessible par des builtins. Une application EDD peut restreindre la taille d'un attribut BLOB mais doit permettre les transferts d'au moins 1 048 575 octets de données.

L'attribut HANDLING doit être READ ou WRITE (et non les deux).

NOTE HANDLING READ signifie que l'attribut BLOB peut uniquement être lu depuis l'équipement.

L'attribut IDENTITY est directement accessible depuis l'EDD à l'aide de la chaîne de référence `blob_reference.IDENTITY`

(qui renvoie une chaîne de caractères correspondant à l'attribut IDENTITY de l'attribut BLOB).

La taille de l'attribut BLOB est directement accessible depuis la DD à l'aide de la référence d'entier

`blob_reference.COUNT`

qui renvoie la taille de l'attribut BLOB (le nombre d'octets).

Structure lexicale

BLOB *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  HANDLING handling-specifier;
  IDENTITY file-name;
}
```

Les attributs BLOB sont spécifiés dans le Tableau 59.

Tableau 59 – Attributs BLOB

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		BLOB	x	x	x	✓	x	x	x
14.6		HANDLING	x	x	x	m	x	x	x
14.8		HELP	x	x	x	o	x	x	x
9.8.2		IDENTITY	x	x	x	o	x	x	x
14.9		LABEL	x	x	x	o	x	x	x

9.8.2 IDENTITY

Objet

L'attribut IDENTITY (nom de fichier de l'attribut BLOB) désigne le fichier informatique ou l'espace de stockage pour un attribut BLOB donné et est utilisé par les applications pour l'identifier. L'attribut IDENTITY est une chaîne contenant jusqu'à 32 caractères ASCII entre guillemets et constituée de lettres (a-z et A-Z), de chiffres (0-9), du caractère de soulignement (_) ou d'un point (.). Afin de s'assurer que l'attribut IDENTITY de l'attribut BLOB ne soit pas confondu avec celui éventuellement utilisé par d'autres équipements, les applications EDD doivent séparer leur espace de stockage pour les attributs BLOB par type d'équipement.

Si IDENTITY n'est pas défini, le nom de fichier doit être choisi au moyen d'une méthode (voir l'IEC 61804-5).

Structure lexicale

IDENTITY *file-name*;

L'attribut IDENTITY est spécifié dans le Tableau 60.

Tableau 60 – Attribut IDENTITY

Type de valeurs	Description
file-name	Littéral de chaîne qui identifie le nom de fichier. Profils de communication: HART Longueur limitée à 32 caractères

10 Communication

10.1 COMMAND

10.1.1 Structure générale

Objet

Un attribut COMMAND est une construction utilisée pour prendre en charge le mapping des éléments de communication EDD vers un système de communication choisi. Un attribut COMMAND spécifie les données transférées vers et depuis un équipement. Les données à écrire sur l'équipement sont spécifiées dans l'attribut REQUEST. Les données à lire depuis l'équipement sont spécifiées dans l'attribut REPLY. Il spécifie différents éléments exigés pour construire une trame de communication.

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, selon le cas. 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 *identifiant*

```
{
  API [integer-constant | reference | integer-expression]<cond>;
  BLOCK_B reference<cond>;
  HEADER string-value<cond>
  INDEX [integer-constant | reference | integer-expression]<cond>;
  NUMBER integer-constant;
  OPERATION [COMMAND | READ | WRITE];
  RESPONSE_CODES { [referenced-response-codes-specifier |
                    response-codes-specifier] }
  SLOT [integer-constant | reference | integer-expression]<cond>;
  SUB_SLOT [integer-constant | reference | integer-expression]<cond>;
  TRANSACTION [integer]?
  {
    REQUEST { request-item-specifier [, request-item-specifier]* }
    REPLY { reply-item-specifier [, reply-item-specifier]* }
    RESPONSE_CODES { [referenced-response-codes-specifier |
                     response-codes-specifier] }
    POST_RECEIVE_ACTIONS { actions-specifier }
  }
}
```

Les attributs COMMAND sont spécifiés dans le Tableau 61.

Tableau 61 – Attributs COMMAND

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	COMMAND		✓	x	✓	✓	x	✓	✓
10.1.3	API		o	x	o	x	x	x	o
10.1.4	BLOCK_B		o	x	x	x	x	o	o
10.1.5	HEADER		o	x	o	x	x	x	x
10.1.6	INDEX		o	x	x	x	x	m	m
10.1.7	NUMBER		o	x	x	m	x	x	x
10.1.8	OPERATION		m	x	m	m	x	m	m
14.15	RESPONSE_CODES	Profils de communication: HART Uniquement: RESPONSE_CODES imbriqués; pas d'instructions conditionnelles. Les attributs COMMAND RESPONSE_CODES sont obligatoires s'il n'existe aucun attribut RESPONSE_CODES dans l'attribut TRANSACTION	o	x	o	c	x	o	o
10.1.9	SLOT		o	x	x	x	x	o	o
10.1.10	SUB_SLOT		o	x	x	x	x	x	o
10.1.11	TRANSACTION	Profils de communication: CS, GPE, PB, PN Un seul attribut TRANSACTION sans définition d'entier	m	x	m	m	x	m	m

NOTE 2 Les codes de réponse apparaissant à l'intérieur d'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.

10.1.2 Adressage de commande

10.1.2.1 Adressage de commande HART

L'attribut NUMBER spécifie le numéro de commande HART utilisé par l'équipement pour identifier la commande et les données.

10.1.2.2 Adressage DP PROFIBUS

Les attributs SLOT et INDEX spécifient l'ensemble de données PROFIBUS transféré. Avec un équipement modulaire, les modules sont généralement adressés avec l'attribut SLOT et les données des modules sont adressées avec l'attribut INDEX.

10.1.2.3 Profil PI pour l'adressage des équipements de contrôle des processus industriels

Les adresses absolues des attributs BLOCK d'un profil PI pour équipements de contrôle des processus sont conservées dans le référentiel de gestion des équipements. Il est nécessaire que l'application EDD extraie le référentiel et résolve l'adresse de base de l'attribut BLOCK (BLOCK_B). L'attribut BLOCK d'un attribut COMMAND est une référence à un élément BLOCK_B.

Un élément EDDL BLOCK_B est décrit avec un attribut TYPE et un attribut NUMBER. L'attribut TYPE définit les types d'un bloc (PHYSICAL, TRANSDUCER ou FUNCTION). L'attribut NUMBER définit le numéro d'un bloc d'un type de blocs dans le référentiel de gestion des équipements. Pour tous les types de blocs, l'attribut NUMBER commence par 1.

L'attribut de commande INDEX est un index relatif, en fonction du premier ensemble de données du bloc. Pour adresser un ensemble de données dans l'équipement, l'emplacement de départ absolu et l'index de départ du référentiel de gestion des équipements sont utilisés, et l'attribut INDEX relatif de l'attribut COMMAND est ajouté.

10.1.2.4 Adressage PROFINET

Les attributs API, SLOT, SUB_SLOT et INDEX sont utilisés pour adresser les ensembles de données dans l'équipement de manière absolue.

10.1.2.5 Adressage pour les autres protocoles

L'attribut de chaîne HEADER permet de contenir les informations de protocole relatives à un attribut COMMAND. L'attribut HEADER n'est pas utilisé avec les équipements PROFIBUS, PROFINET, FOUNDATION Fieldbus, ISA100_Wireless et HART. Le contenu concret de la chaîne ne relève pas du domaine d'application du présent document.

10.1.3 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 non limitée.

Structure lexicale

API [*integer-constant* | *reference* | *integer-expression*]<cond>;

Les attributs API sont spécifiés dans le Tableau 62.

Tableau 62 – Attribut API

Attribut	Description
integer-constant	Numéro d'API.
reference	Référence à une instance de VARIABLE contenant le numéro d'API.
integer-expression	Expression qui doit être évaluée à un entier non négatif pour le numéro d'API (pour la description des expressions, voir Article 18).

10.1.4 BLOCK_B

Objet

L'attribut BLOCK_B fournit une référence à un bloc individuel dans l'équipement.

Structure lexicale

BLOCK_B *reference*<cond>;

L'attribut BLOCK_B est spécifié dans le Tableau 63.

Tableau 63 – Attribut BLOCK_B

Type de valeurs	Description
reference	Référence à une instance de BLOCK_B.

10.1.5 HEADER

Objet

L'attribut HEADER est utilisé pour des protocoles de communication spécifiques.

Il peut être utilisé pour adresser des données dans un équipement ou pour une opération de réinitialisation.

Structure lexicale

HEADER *string-value*<cond>

L'attribut HEADER est spécifié dans le Tableau 64.

Tableau 64 – Attribut HEADER

Attribut	Description
string-value	Chaîne qui est utilisée pour la communication avec des protocoles de communication spécifiques.

10.1.6 INDEX

Objet

L'attribut INDEX doit être présent si l'attribut SLOT ou BLOCK_B est utilisé. L'attribut INDEX spécifie l'adresse de l'élément de données dans l'attribut SLOT ou BLOCK_B.

Structure lexicale

INDEX [*integer-constant* | *reference* | *integer-expression*]<cond>;

Les attributs INDEX sont spécifiés dans le Tableau 65.

Tableau 65 – Attribut INDEX

Type de valeurs	Description
integer-constant	Numéro de l'index.
reference	Référence à une instance de VARIABLE contenant l'index.
integer-expression	Expression qui doit être évaluée à un entier non négatif qui se situe dans la plage d'index de l'attribut INDEX (pour la description des expressions, voir Article 18).

10.1.7 NUMBER

Objet

L'attribut NUMBER est utilisé pour spécifier le numéro de l'attribut COMMAND.

Structure lexicale

NUMBER *unsigned-integer-constant*;

L'attribut NUMBER est spécifié dans le Tableau 66.

Tableau 66 – Attribut NUMBER

Attribut	Description
unsigned-integer-constant	Numéro de commande HART

10.1.8 OPERATION

Objet

L'attribut OPERATION spécifie les actions effectuées par l'équipement lors de la réception d'une demande de service provenant du système de communication.

L'attribut OPERATION READ spécifie que l'objectif principal de cet attribut COMMAND est de lire les éléments de données depuis l'équipement. Toutefois, en fonction du protocole ou des mécanismes d'adressage propres à l'équipement, une commande READ peut spécifier les données à transférer vers l'équipement (par exemple, les variables INFO et INDEX avant la réception des données en provenance de l'équipement). Avec ces données, l'équipement peut adresser les éléments de données qu'il convient de transférer vers l'application EDD.

L'attribut OPERATION WRITE spécifie que l'attribut COMMAND écrit les éléments de données sur l'équipement. Une commande WRITE peut spécifier les données dans l'attribut REPLY transféré depuis l'équipement après écriture. Si l'attribut REPLY comporte des éléments de données qui sont écrits sur l'équipement avec cette commande, les modifications de données apportées par l'équipement peuvent être détectées dans l'application EDD.

L'attribut OPERATION COMMAND spécifie que l'objectif de l'attribut COMMAND est d'activer une fonction. L'attribut REQUEST spécifie les paramètres d'entrée. L'attribut REPLY spécifie les paramètres de sortie. L'application EDD ne doit pas utiliser ces attributs COMMAND pour le transfert de données. Ces attributs COMMAND peuvent uniquement être utilisés avec les builtins de communication dans les attributs METHOD.

Structure lexicale

OPERATION [COMMAND | READ | WRITE];

Les attributs OPERATION sont spécifiés dans le Tableau 67.

Tableau 67 – Attribut OPERATION

Valeur d'attribut	Description
COMMAND	L'équipement effectue un ensemble prédéfini d'actions (par exemple, auto-essai, réinitialisation principale).
READ	Spécifie une transaction de lecture. L'équipement retourne les valeurs actuelles des variables spécifiées. VARIABLES doit être défini dans la section REPLY de TRANSACTION.
WRITE	Spécifie une transaction d'écriture. L'équipement reçoit les valeurs actuelles des variables spécifiées et définit les valeurs du jeu de variables spécifié. VARIABLES doit être défini dans la section REQUEST de TRANSACTION.

Le Tableau 68 définit le mapping des communications vers les services de communication pour PROFIBUS et PROFINET.

Tableau 68 – Mapping des communications PROFIBUS et PROFINET

Eléments REQUEST	Eléments REPLY	Commande
non	non	L'application EDD doit utiliser un service d'écriture
oui	non	L'application EDD doit utiliser un service d'écriture
non	oui	L'application EDD doit utiliser un service de lecture
oui	oui	Non admise. L'application EDD doit générer une erreur et n'effectuer aucune communication.

10.1.9 SLOT

Objet

L'attribut SLOT spécifie le numéro d'un emplacement. Les éléments de données d'un équipement peuvent être alloués à des emplacements logiques. Dans un emplacement, chaque élément de données est adressé en utilisant un index.

Structure lexicale

SLOT [*integer-constant* | *reference* | *integer-expression*]<cond>;

Les attributs SLOT sont spécifiés dans le Tableau 69.

Tableau 69 – Attribut SLOT

Type de valeurs	Description
integer-constant	Numéro d'emplacement.
reference	Référence à une instance de VARIABLE contenant le numéro d'emplacement.
integer-expression	Expression qui doit être évaluée à un entier non négatif qui se situe dans la plage d'index de l'attribut SLOT (pour la description des expressions, voir Article 18).

10.1.10 SUB_SLOT

Objet

L'attribut SUB_SLOT spécifie le numéro d'un sous-emplacement. Les éléments de données d'un équipement peuvent être alloués à des sous-emplacements logiques. Dans un sous-emplacement, chaque élément de données est adressé en utilisant un index.

Structure lexicale

SUB_SLOT [*integer-constant* | *reference* | *integer-expression*]<cond>;

Les attributs SUB_SLOT sont spécifiés dans le Tableau 70.

Tableau 70 – Attribut SUB_SLOT

Type de valeurs	Description
integer-constant	Numéro de sous-emplacement.
reference	Référence à une instance de VARIABLE contenant le numéro de sous-emplacement.
integer-expression	Expression qui doit être évaluée à un entier non négatif qui se situe dans la plage d'index de l'attribut SUB_SLOT (pour la description des expressions, voir Article 18).

10.1.11 TRANSACTION

10.1.11.1 Structure générale

Objet

Les transactions spécifient le champ de données des messages de demande et de réponse d'une commande. Il est possible de définir plusieurs transactions pour le profil de communication HART. Lorsqu'une commande comporte plusieurs transactions, chaque transaction doit être marquée par un numéro après les mots-clés de l'attribut TRANSACTION de manière à pouvoir distinguer les différentes transactions. Lorsque le numéro de transaction n'est pas spécifié, l'hypothèse retenue est qu'il est égal à zéro. Chaque attribut TRANSACTION peut comprendre un ensemble de codes de réponse qui s'applique seulement à cette transaction particulière.

Structure lexicale

TRANSACTION [*integer*]?

```
{
  REQUEST { request-item-specifier [, request-item-specifier]* }
  REPLY { reply-item-specifier [, reply-item-specifier]* }
  RESPONSE_CODES { [referenced-response-codes-specifier |
                    response-codes-specifier] }
}
```

Les attributs TRANSACTION sont spécifiés dans le Tableau 71.

Tableau 71 – Attributs TRANSACTION

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
	integer	o	x	o	o	x	o	o
10.1.11.2	REPLY Profils de communication: HART Pas d'instructions conditionnelles. L'attribut REPLY peut uniquement contenir des références à l'attribut VARIABLE; les instances intégrales des attributs LIST, VALUE_ARRAY, REFERENCE_ARRAY et COLLECTION ne sont pas prises en charge.	m	x	m	m	x	m	m
10.1.11.3	REQUEST Profils de communication: HART Pas d'instructions conditionnelles. L'attribut REQUEST peut uniquement contenir des références à l'attribut VARIABLE; les instances intégrales des attributs LIST, VALUE_ARRAY, REFERENCE_ARRAY et COLLECTION ne sont pas prises en charge.	m	x	m	m	x	m	m
14.15	RESPONSE_CODES Profils de communication: HART Uniquement: RESPONSE_CODES imbriqués; pas d'instructions conditionnelles. Les attributs TRANSACTION RESPONSE_CODES sont obligatoires s'il n'existe aucun attribut RESPONSE_CODES dans l'attribut COMMAND	o	x	o	c	x	o	o
10.1.11.6	POST_RECEIVE_ACTIONS	x	x	x	o	x	x	x

10.1.11.2 REPLY

Objet

L'attribut REPLY contient une liste d'éléments de données qui sont reçus par l'équipement. L'ordre de la liste d'éléments de données doit être maintenu dans le champ de données correspondant dans le PDU communiqué. Si les éléments de données référencés 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 des attributs LIST, VALUE_ARRAY, REFERENCE_ARRAY et COLLECTION avec un attribut item-mask n'est pas admise.

Lorsqu'un élément de données ne commence et ne termine pas sur des limites d'octet, il convient d'utiliser un attribut item-mask pour spécifier comment la variable est extraite du champ de données. Même si le bit de poids faible n'est pas utilisé pour des données réelles, il convient de le définir dans un attribut item-mask (associé à un élément de données "fictif") si des éléments de données suivent.

Lorsqu'une instance intégrale de l'attribut LIST est référencée, l'application EDD doit réinitialiser l'attribut LIST, puis renseigner l'attribut LIST avec les données du PDU reçu.

Structure lexicale

```
REPLY { reply-item-spezifier [, reply-item-spezifier]* }
reply-item-spezifier:= [integer-constant | [reference [< item-mask >]]?
[INDEX | INFO][?]]
```

Les attributs REPLY sont spécifiés dans le Tableau 72.

Tableau 72 – Attributs REPLY et REQUEST

Attribut	Description
integer	Entier littéral qui affiche toujours une taille de un octet.
reference	Référence à une instance de l'attribut VARIABLE Profils de communication: PB, PN, CS, GPE En outre, référence à une instance de l'attribut VALUE_ARRAY, COLLECTION, REFERENCE_ARRAY ou LIST.
item-mask	Motif de bits pour extraire l'attribut VARIABLE du champ de données (voir 10.1.11.4). L'attribut item-mask doit être utilisé uniquement pour les types de données INTEGER, UNSIGNED INTEGER, ENUMERATED et BIT_ENUMERATED. La longueur de l'attribut masque-d-élément doit correspondre à la longueur de l'élément de données dans la liste REQUEST ou REPLY. Profils de communication: HART Aucun espace ne doit être utilisé entre les masques d'élément.
INFO	L'attribut VARIABLE n'est pas réellement stocké dans l'équipement ou l'application EDD. Il contient des informations supplémentaires pour assurer un traitement correct.
INDEX	L'attribut VARIABLE est utilisé dans les attributs REQUEST ou REPLY en tant qu'index des attributs REFERENCE_ARRAY, LIST et VALUE_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 l'attribut INDEX, à l'exception que les variables d'index ne sont pas stockées dans l'équipement ou l'application EDD.

L'attribut INFO peut être utilisé pour envoyer ou lire un attribut VARIABLE avec des unités différentes de celles utilisées dans l'équipement.

10.1.11.3 REQUEST

Objet

L'attribut REQUEST contient une liste d'éléments de données qui sont envoyés à l'équipement. L'ordre de la liste d'éléments de données doit être maintenu dans le champ de données correspondant dans le PDU communiqué. Si les éléments de données référencés 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 des attributs LIST, VALUE_ARRAY, REFERENCE_ARRAY et COLLECTION avec un attribut item-mask n'est pas admise.

Lorsqu'un élément de données 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.

Même si le bit de poids faible n'est pas utilisé pour des données réelles, il convient de le définir dans un attribut item-mask (associé à un élément de données "fictif") si des éléments de données suivent.

Structure lexicale

```
REQUEST { request-item-specifier [, request-item-specifier]* }
request-item-specifier:= [integer-constant | [reference [< item-mask >]]?
[INDEX | INFO]?]]
```

Les attributs REQUEST sont spécifiés dans le Tableau 72.

10.1.11.4 item-mask

Objet

Les masques d'élément sont uniquement nécessaires lorsque les attributs VARIABLE ou les constantes ne commencent ni ne se terminent sur des limites d'octet. Un élément sans masque d'élément commence sur le bit 0. Les masques d'élément peuvent être spécifiés pour les attributs VARIABLE et les constantes. Les bits avec la valeur "1" dans le masque d'élément définissent le mapping de l'élément de données vers la trame de données. Les bits avec la valeur "1" doivent être consécutifs. Si la valeur du bit 0 du masque d'élément est "1", l'élément suivant est associé par mapping au bit 0 de l'octet suivant dans la trame de données. Les bits non contenus dans le masque d'élément sont définis sur 0. Le chevauchement des masques d'élément n'est admis que si le bit 0 du masque précédent est défini.

Dans un paquet de réponses,

- le flux de bits de commande est masqué (de bit à bit) avec la valeur du masque d'élément;
- il est décalé à droite du nombre de bits "0" de fin consécutifs dans le masque d'élément;
- les bits correspondants de l'élément de données sont définis sur la valeur résultante; et
- les autres bits de la variable de destination ne sont pas affectés.

Dans un paquet de demandes,

- l'élément de données est décalé à gauche du nombre de bits "0" de fin consécutifs dans le masque d'élément;
- le résultat est masqué (de bit à bit) avec la valeur du masque d'élément; et
- ce résultat est inséré dans les bits correspondants des octets actuellement actifs du paquet.

Chaque bit dans le masque peut avoir une signification sémantique. De plus, si le bit de poids faible est défini dans un attribut item-mask, l'élément de données suivant (le cas échéant) est contenu dans le ou les octets suivants du PDU. Si le bit de poids faible est nul (non défini), l'élément de données suivant est contenu dans le ou les mêmes octets du PDU.

Les masques d'élément multioctets définissent un mapping vers une plage plus vaste de la trame de données. Le nombre de bits avec la valeur "1" dans le masque d'élément doit être inférieur ou égal au nombre de bits de l'élément de données.

Si un masque d'élément est spécifié pour un attribut VARIABLE ou une constante, et que les éléments précédent et suivant n'ont pas de masque d'élément, le premier bit défini dans le masque détermine la position de départ et le dernier bit défini détermine la position de fin dans la trame de données.

Les masques doivent être spécifiés du MSB au LSB dans la trame de données.

Profils de communication: HART

Les espaces ne sont pas admis au sein d'un masque d'élément.

Utilisation

La Figure 33 représente l'utilisation des masques d'élément de données.

```
REQUEST {
  x <0xF8>, y <0x07>
}
```

Figure 33 – Exemple de masque d'élément

Le champ de données de la demande comporte deux attributs VARIABLE qui sont encapsulés dans un octet. L'attribut VARIABLE x est encapsulé dans les cinq premiers bits de poids fort et l'attribut VARIABLE y est encapsulé dans les trois premiers bits de poids faible. Si x était égal à 155 et y à 5, l'octet transmis serait 221 (0xDD).

```
COMMAND cmd_abc
{
  OPERATION WRITE; // ou READ
  ...             // attributs spécifiques au protocole, par exemple, SLOT et INDEX
  TRANSACTION
  {
    REQUEST        // ou REPLY avec READ
    {
      a,
      b <0x3c>,    // 00111100b
      c
    }
  }
}
```

Figure 34 – Exemple de masque d'élément simple

Sur la Figure 34, a, b et c sont des attributs VARIABLE INTEGER(1). Dans ce cas, les bits 0 à 3 de l'attribut VARIABLE b sont associés par mapping aux bits 2 à 5 de la trame de données derrière l'attribut VARIABLE a. L'attribut VARIABLE c commence sur le bit 0 de l'octet suivant de la trame de données (voir Figure 35).

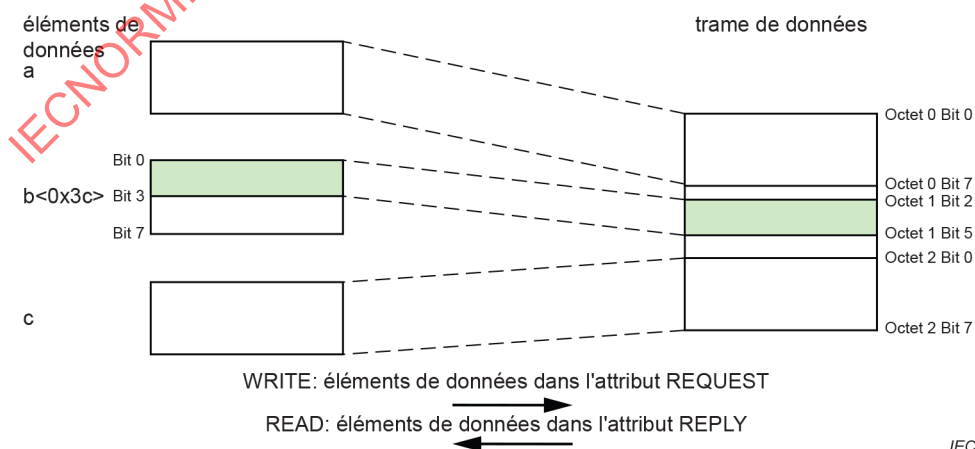


Figure 35 – Exemple de mapping avec un masque d'élément simple

Sur la Figure 36, a, c, d et e sont des attributs VARIABLE INTEGER(1), et b est un attribut VARIABLE INTEGER(2). Dans ce cas, les Bits 0 à 11 de l'attribut VARIABLE b sont associés par mapping au Bit 0 de l'Octet 1 et jusqu'au Bit 3 de l'Octet 2 de la trame de données. Les

Bits 0 à 1 de l'attribut VARIABLE c sont associés par mapping aux Bits 4 à 5 de l'Octet 2 de la trame de données. Le Bit 0 de l'attribut VARIABLE d est associé par mapping au Bit 7 de l'Octet 2 (voir Figure 37).

```

COMMAND cmd_abcde
{
  OPERATION WRITE; // ou READ
  ... // attributs spécifiques au protocole, par exemple, SLOT et INDEX
  TRANSACTION
  {
    REQUEST // ou REPLY avec READ
    {
      a,
      b <0xFFF0>, // 1111111111110000b
      c <0x000C>, // 0000000000001100b
      0 <0x0002>, // 000000000000010b, exigé uniquement avec HART, aucun espace
      admis
      d <0x0001>, // 0000000000000001b
      e
    }
  }
}

```

Figure 36 – Exemple de masque d'élément multiple

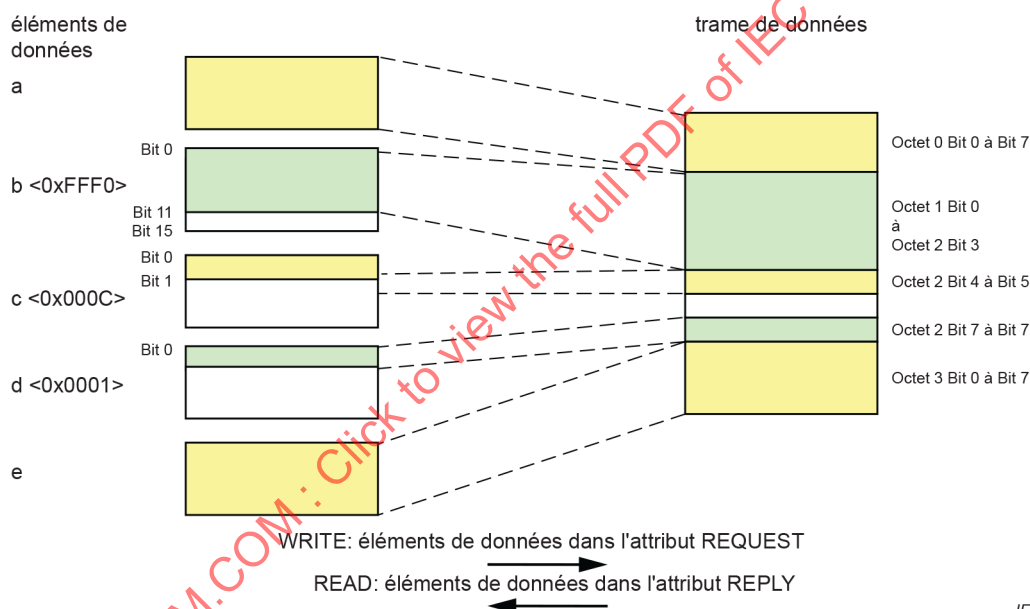


Figure 37 – Exemple de mapping avec un masque d'élément multiple

10.1.11.5 Qualificatifs d'élément de données

Objet

Un attribut VARIABLE apparaissant dans une demande ou une réponse peut être qualifié par les mots-clés INDEX et INFO.

Un attribut VARIABLE qualifié par un attribut INDEX spécifie que l'attribut VARIABLE est utilisé dans la demande ou la réponse comme un index matriciel.

Un attribut VARIABLE qualifié par un attribut INFO spécifie que la variable n'est pas réellement stockée dans l'équipement ou l'application EDD; l'attribut VARIABLE est uniquement communiqué à des fins d'information.

Un attribut VARIABLE peut être qualifié à la fois par les attributs INDEX et INFO; dans ce cas, l'attribut VARIABLE est utilisé comme un index matriciel et n'est pas réellement stocké dans l'équipement.

Utilisation

La Figure 38 représente un exemple d'EDD démontrant l'utilisation d'un attribut VARIABLE qualifié par un attribut INFO.

```
TRANSACTION
{
  REQUEST
  {
    units (INFO), x, y
  }
  REPLY
  {
    units, x, y
  }
}
```

Figure 38 – Qualificatif INFO

Les attributs VARIABLE x et y sont écrits sur l'équipement dans l'unité spécifiée par "units". Les attributs VARIABLE x et y sont stockées dans l'équipement (vraisemblablement dans une unité fixe), mais "units" n'est pas défini. Cela permet de communiquer un attribut VARIABLE dans une unité différente de l'unité actuelle (celle dans laquelle la variable est affichée).

La Figure 39 représente un exemple d'EDD démontrant l'utilisation d'un attribut VARIABLE qualifié par un attribut INDEX.

```
TRANSACTION
{
  REQUEST
  {
    code (INDEX), table[code]
  }
  REPLY
  {
    code, table[code]
  }
}
```

Figure 39 – Qualificatif INDEX

L'élément de données dans le tableau matriciel est spécifié par un code et écrit sur l'équipement. L'équipement conserve la valeur du code et répète sa valeur. L'élément de données envoyé avec le code est également conservé et la valeur stockée dans l'équipement est renvoyée.

La Figure 40 représente un exemple démontrant l'utilisation d'attributs VARIABLE d'index local.

```
TRANSACTION
{
  REQUEST
  {
    code (INFO, INDEX), table[code]
  }
  REPLY
  {
    code, table[code]
  }
}
```

Figure 40 – Qualificatifs INFO et INDEX

Le "code" des attributs VARIABLE avec les qualificatifs INFO et INDEX est envoyé vers l'équipement, mais n'est pas stocké dans l'équipement ou l'application EDD. L'attribut VARIABLE est uniquement utilisé comme index pour le tableau matriciel.

10.1.11.6 POST_RQSTRECEIVE_ACTIONS

Objet

POST_RQSTRECEIVE_ACTIONS ne doit être pris en charge que par les simulateurs d'équipement.

L'attribut POST_RQSTUPDATE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés par le simulateur d'équipement lorsque 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 la génération de la section RÉPLY. 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<cond> [, reference<cond>]*
}
```

L'attribut POST_RQSTRECEIVE_ACTIONS est spécifié dans le Tableau 73.

Tableau 73 – Attributs POST_RQSTRECEIVE_ACTIONS

Type de valeurs	Description
reference	Référence à un attribut METHOD.

10.2 RECORD

Objet

Un attribut RECORD est un groupe logique d'attributs VARIABLE utilisés pour représenter un objet de communication. Chaque attribut MEMBER dans l'attribut RECORD est une référence à un attribut VARIABLE et peut être constitué de différents types de données. Un attribut VARIABLE peut être référencé depuis un autre emplacement dans l'EDD par l'identificateur de RECORD et l'identificateur de MEMBER.

Structure lexicale

RECORD *identifiant*

```
{
  LABEL string-value;
  HELP string-value;
  MEMBERS { members-specifier }
  PRIVATE boolean-specifier;
  RESPONSE_CODES referenced-response-codes-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  WRITE_MODE write-mode-specifier;
}
```

Les attributs RECORD sont spécifiés dans le Tableau 74.

Tableau 74 – Attributs RECORD

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	RECORD		x	✓	x	x	✓	x	x
14.8	HELP		x	o	x	x	o	x	x
14.9	LABEL		x	m	x	x	m	x	x
14.12	MEMBERS		x	m	x	x	m	x	x
14.13	PRIVATE		x	o	x	x	o	x	x
14.15	RESPONSE_CODES		x	o	x	x	o	x	x
		Profils de communication: FF, ISA100 Uniquement les attributs RESPONSE_CODES référencés.							
14.17	VALIDITY		x	o	x	x	o	x	x
14.18	VISIBILITY		x	o	x	x	o	x	x
14.20	WRITE_MODE		x	o	x	x	o	x	x

10.3 RESPONSE_CODES

Objet

Les attributs RESPONSE_CODES spécifient les valeurs qu'un équipement peut retourner en tant qu'information d'erreur. Chaque attribut VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST ou COMMAND peut avoir son propre ensemble d'attributs RESPONSE_CODES.

Structure lexicale

```
RESPONSE_CODES identifier
{
  [[integer , response-code-type , description , [help]?]<cond>;]+
}
response-code-type:= [DATA_ENTRY_ERROR | DATA_ENTRY_WARNING | MISC_ERROR |
MISC_WARNING | MODE_ERROR | PROCESS_ERROR | SUCCESS]
```

L'attribut RESPONSE_CODES est spécifié dans le Tableau 75.

Tableau 75 – Présentation de l'attribut RESPONSE_CODES

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	RESPONSE_CODES		✓	✓	✓	x	✓	✓	✓

Les attributs RESPONSE_CODES sont spécifiés dans le Tableau 76.

Tableau 76 – Attributs RESPONSE_CODES

Attribut	Description
description	Chaîne qui s'affiche lorsque le code de réponse est retourné par l'équipement.
integer	Valeur retournée de l'équipement pour identifier le code de réponse.
DATA_ENTRY_ERROR	Le service de couche application a été rejeté parce que les données envoyées ne sont pas adaptées.
DATA_ENTRY_WARNING	Le service de couche application a été accepté et traité avec une version des données envoyées légèrement modifiée.
MISC_ERROR	Le service de couche application a été rejeté.
MISC_WARNING	Le service de couche application a été accepté et traité comme spécifié et il existe des informations supplémentaires, sans rapport avec le service de couche application, qui pourraient intéresser l'utilisateur.
MODE_ERROR	Le service de couche application a été rejeté parce que l'équipement était dans un mode dans lequel le service de couche application ne pouvait pas être exécuté.
PROCESS_ERROR	Le service de couche application a été rejeté parce qu'un processus appliqué à l'équipement n'est pas adapté.
SUCCESS	Le service de couche application a été accepté et traité comme spécifié.
help	Texte d'aide pour l'élément.

10.4 VARIABLE_LIST

Objet

Un attribut VARIABLE_LIST est un groupe d'objets de communication EDD (VARIABLE, VALUE_ARRAY ou RECORDS). Les objets de communication peuvent être référencés depuis un autre emplacement dans l'EDD par l'identificateur de VARIABLE_LIST et l'identificateur de MEMBER. Les attributs VARIABLE_LIST sont utilisés pour regrouper des objets afin de faciliter l'exécution des applications. Les attributs VARIABLE_LIST permettent d'optimiser la communication en offrant la possibilité de regrouper plusieurs éléments dans un nombre réduit d'objets de communication.

Structure lexicale

```
VARIABLE_LIST identifier
{
    LABEL string-value;
    HELP string-value;
    MEMBERS { members-specifier }
    RESPONSE_CODES referenced-response-codes-specifier;
}
```

Les attributs VARIABLE_LIST sont spécifiés dans le Tableau 77.

Tableau 77 – Attributs VARIABLE_LIST

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		VARIABLE_LIST	x	✓	x	x	✓	x	x
14.8		HELP	x	o	x	x	o	x	x
14.9		LABEL	x	o	x	x	o	x	x
14.12		MEMBERS	x	m	x	x	m	x	x
14.15		RESPONSE_CODES	x	o	x	x	o	x	x

11 Interface utilisateur

11.1 MENU

11.1.1 Structure générale

Objet

L'attribut MENU organise dans une structure hiérarchique des variables, des méthodes et d'autres éléments spécifiés dans le langage EDDL. Les attributs MENU constituent le mécanisme principal pour organiser les informations de l'équipement et prendre en charge les interactions de l'utilisateur avec l'équipement. Une application utilisateur peut utiliser les attributs MENU ITEMS pour afficher et entrer des informations d'une façon organisée et cohérente. La hiérarchie d'un attribut MENU est créée en définissant un menu et en l'intégrant comme un élément d'un autre menu.

Structure lexicale

MENU *identifiant*

```
{
  LABEL string-value;
  HELP string-value;
  ACCESS [OFFLINE | ONLINE];
  EXIT_ACTIONS { actions-spezififier }
  INIT_ACTIONS { actions-spezififier }
  ITEMS { menu-item-spezififier [, menu-item-spezififier]* }
  POST_EDIT_ACTIONS { actions-spezififier }
  POST_READ_ACTIONS { actions-spezififier }
  POST_WRITE_ACTIONS { actions-spezififier }
  PRE_EDIT_ACTIONS { actions-spezififier }
  PRE_READ_ACTIONS { actions-spezififier }
  PRE_WRITE_ACTIONS { actions-spezififier }
  STYLE [DIALOG | WINDOW | PAGE | GROUP | MENU | TABLE];
  VALIDITY boolean-spezififier;
  VISIBILITY boolean-spezififier;
}
```

Les attributs MENU sont spécifiés dans le Tableau 78.

Tableau 78 – Attributs MENU

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	MENU		✓	✓	✓	✓	✓	✓	✓
11.1.2	ACCESS		o	x	o	x	x	o	o
11.1.3	EXIT_ACTIONS		o	o	o	x	o	o	o
14.8	HELP		o	o	o	o	o	o	o
11.1.4	INIT_ACTIONS		o	o	o	x	o	o	o
11.1.5	ITEMS		m	m	m	m	m	m	m
14.9	LABEL		m	m	m	m	m	m	m
11.1.6	POST_EDIT_ACTIONS		o	x	o	x	x	o	o
11.1.7	POST_READ_ACTIONS		o	o	o	L ^a	o	o	o
11.1.8	POST_WRITE_ACTIONS		o	o	o	L ^a	o	o	o
11.1.9	PRE_EDIT_ACTIONS		o	x	o	x	x	o	o
11.1.10	PRE_READ_ACTIONS		o	o	o	L ^a	o	o	o
11.1.11	PRE_WRITE_ACTIONS		o	o	o	L ^a	o	o	o
11.1.12	STYLE		o	o	o	o	o	o	o
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	o	o	o	o
Les attributs PRE_READ_ACTIONS et POST_READ_ACTIONS doivent uniquement être utilisés lors du téléchargement montant à partir des menus d'un équipement. Les attributs PRE_WRITE_ACTIONS et POST_WRITE_ACTIONS doivent uniquement être utilisés lors du téléchargement descendant vers les menus d'un équipement. ^a Profils de communication: HART									

11.1.2 ACCESS

Objet

L'attribut ACCESS spécifie si les éléments affichés ou utilisés dans l'attribut MENU sont lus depuis l'équipement (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 79.

Tableau 79 – Attribut ACCESS

Valeur d'attribut	Description
OFFLINE	Les valeurs des éléments affichés dans l'attribut MENU sont lues localement.
ONLINE	Il convient de lire les valeurs des éléments affichés dans l'attribut MENU depuis l'équipement et de les actualiser selon un intervalle approprié.

NOTE La mise en œuvre est responsable de la décision du moment auquel le transfert de valeurs modifiées vers un équipement est effectué (généralement après l'exécution d'une entrée unique ou d'un attribut MENU pour un attribut ONLINE ACCESS ou par un menu séparé pour mettre à jour l'équipement).

11.1.3 EXIT_ACTIONS

Objet

L'attribut EXIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés lorsque l'utilisateur a terminé le traitement de l'attribut MENU, même si ce dernier 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 d'un attribut MENU doivent être exécutés après tous les autres attributs ACTIONS de l'attribut MENU.

Structure lexicale

```
EXIT_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs EXIT_ACTIONS sont spécifiés dans le Tableau 82.

11.1.4 INIT_ACTIONS

Objet

L'attribut INIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement après l'activation de l'attribut 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 d'un attribut MENU doivent être exécutés avant tous les autres attributs ACTIONS de l'attribut MENU.

Structure lexicale

```
INIT_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs INIT_ACTIONS sont spécifiés dans le Tableau 82.

11.1.5 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 des attributs MENU sont présentés à l'utilisateur dans l'ordre dans lequel ils apparaissent.

NOTE 2 Pour les menus verticaux, le premier élément est affiché en haut et le dernier élément en bas. Pour les menus horizontaux, le premier élément est affiché sur la gauche et le dernier élément sur la droite.

Structure lexicale

ITEMS

```

{
    menu-item-spezifier [, menu-item-spezifier]*
}
menu-item-spezifier:=
[menu-reference [(REVIEW)]?]<cond> |
[variable-reference [(DISPLAY_VALUE | HIDDEN | READ_ONLY | NO_LABEL |
NO_UNIT)]?]<cond> |
[image-reference [(INLINE | NO_LABEL | ALIGN_LEFT | ALIGN_RIGHT)]?]<cond>
|
reference<cond> |
string-constant<cond> |
SEPARATOR<cond> |
ROWBREAK |
COLUMNBREAK

```

Les attributs MENU ITEMS sont spécifiés dans le Tableau 80.

Les qualificatifs MENU ITEMS sont spécifiés dans le Tableau 81.

Tableau 80 – Attributs MENU ITEMS

Attribut	Description
menu-reference	Référence à un attribut MENU.
variable-reference	Référence à un attribut VARIABLE.
image-reference	Référence à un attribut IMAGE.
reference	Référence à un bit d'un attribut VARIABLE BIT_ENUMERATED, à un attribut BLOCK_A PARAMETERS, à un élément d'un attribut BLOCK_A PARAMETERS, à un attribut CHART, à un attribut COLLECTION, à un membre d'un attribut COLLECTION, à un attribut EDIT_DISPLAY, à un attribut GRAPH, à un attribut GRID, à un attribut LIST, à un membre d'un attribut LIST, à un attribut METHOD, à un attribut METHOD avec arguments, à un attribut PLUGIN, à un attribut RECORD, à un attribut REFERENCE_ARRAY, à un membre d'un attribut REFERENCE_ARRAY, à un attribut VALUE_ARRAY, ou à un membre d'un attribut VALUE_ARRAY. Profils de communication: HART Les références à un seul élément d'un attribut LIST, REFERENCE_ARRAY et VALUE_ARRAY sont admises. L'attribut BLOCK_A PARAMETERS, l'élément d'un attribut BLOCK_A PARAMETERS, l'attribut LIST, l'attribut METHOD avec arguments, l'attribut RECORD, l'attribut REFERENCE_ARRAY et l'attribut VALUE_ARRAY ne sont pas admis.
SEPARATOR	Il convient que les éléments de menu avant et après l'attribut SEPARATOR soient séparés. NOTE Il est possible que les applications EDD ne puissent pas distinguer un attribut SEPARATOR d'un attribut COLUMNBREAK. L'un ou l'autre de ces attributs est généralement utilisé en fonction de l'attribut STYLE de l'attribut MENU parent.
COLUMNBREAK	Spécifie qu'il convient de commencer une nouvelle colonne.
ROWBREAK	Spécifie qu'il convient de commencer une nouvelle ligne.
string-constant	Texte statique qu'il convient d'afficher sur le menu.

Pour en savoir plus sur les conteneurs de menu et les éléments contenus, voir l'IEC 61804-4.

Tableau 81 – Qualificatifs MENU ITEMS

Attribut	Description
DISPLAY_VALUE	Il convient d'afficher la valeur de la variable avec son libellé (applications EDD à affichage limité). Le qualificatif DISPLAY_VALUE doit faire l'objet d'une hypothèse et être appliqué à tous les attributs VARIABLE affichés par les applications EDD (applications EDD à affichage non limité).
HIDDEN	Profils de communication: CS, GPE, PB, PN Doit uniquement être utilisé avec des éléments de données. L'élément de données doit être lu, mais il n'est pas affiché. Aucun espace ne doit être réservé.
READ_ONLY	Ne doit être utilisé qu'avec VARIABLE. Spécifie que la valeur doit être affichée, mais qu'elle ne doit pas être modifiable depuis l'attribut MENU (indépendamment de son attribut HANDLING).
NO_LABEL	Spécifie que le libellé de l'élément n'est pas affiché; l'attribut DISPLAY_VALUE est implicite. Profils de communication: CS, GPE, FF, PB, PN, ISA100 Ne doit être utilisé qu'avec VARIABLE.
NO_UNIT	Ne doit être utilisé qu'avec VARIABLE. Spécifie que l'unité technique de l'attribut VARIABLE ne doit pas être affichée; l'attribut DISPLAY_VALUE est implicite.
REVIEW	Ne doit être utilisé qu'avec MENU. L'attribut REVIEW est utilisé pour présenter une grande quantité de données, à des fins de révision, par exemple. Seuls les attributs variable-référence ou menu-référence doivent être inclus dans la liste d'attributs ITEMS pour les menus qualifiés avec l'attribut REVIEW. Toutes les variables qui apparaissent sur le menu ou ses sous-menus doivent prendre pour hypothèse les qualificatifs READ_ONLY et DISPLAY_VALUE.
INLINE	Ne doit être utilisé qu'avec IMAGE. Si l'attribut IMAGE est dans un attribut MENU avec l'attribut STYLE égal à WINDOW, DIALOG, PAGE ou GROUP et si le qualificatif INLINE n'est pas spécifié, il convient que l'attribut IMAGE couvre l'étendue de l'attribut MENU; sinon, il convient que l'attribut IMAGE soit affiché en ligne avec les autres attributs ITEMS de l'attribut MENU. Le libellé d'un élément ne doit pas être affiché.
ALIGN_LEFT	Ne doit être utilisé qu'avec IMAGE. Si l'attribut IMAGE est dans un attribut MENU avec l'attribut STYLE égal à WINDOW, DIALOG, PAGE ou GROUP et si le qualificatif ALIGN_LEFT est spécifié, il convient que l'attribut IMAGE s'affiche aligné à gauche sur la ligne ou sur la colonne en fonction du qualificatif INLINE. Si aucun alignement n'est spécifié, il convient que l'attribut IMAGE s'affiche centré sur la ligne ou sur la colonne en fonction du qualificatif INLINE.
ALIGN_RIGHT	Ne doit être utilisé qu'avec IMAGE. Si l'attribut IMAGE est dans un attribut MENU avec l'attribut STYLE égal à WINDOW, DIALOG, PAGE ou GROUP et si le qualificatif ALIGN_RIGHT est spécifié, il convient que l'attribut IMAGE s'affiche aligné à droite sur la ligne ou sur la colonne en fonction du qualificatif INLINE. Si aucun alignement n'est spécifié, il convient que l'attribut IMAGE s'affiche centré sur la ligne ou sur la colonne en fonction du qualificatif INLINE.

11.1.6 POST_EDIT_ACTIONS**Objet**

L'attribut POST_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés lorsque l'utilisateur a terminé le traitement de l'attribut 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 l'attribut MENU sont exécutés uniquement lorsque tous les attributs POST_EDIT_ACTIONS spécifiques à l'entité sont terminés.

Structure lexicale

```
POST_EDIT_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
[DEFINITION { compound-statement }]<cond>?
```

Les attributs POST_EDIT_ACTIONS sont spécifiés dans le Tableau 82.

Tableau 82 – Attributs EXIT_ACTIONS, INIT_ACTIONS, POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS

Type de valeurs	Description
reference	Profils de communication: FF, HART, ISA100 Référence à une instance de METHOD. Profils de communication: CS, GPE, PB, PN Référence à une instance de l'attribut METHOD avec arguments.
DEFINITION	Profils de communication: CS, GPE, PB, PN Élément lexical (voir 14.4).

11.1.7 POST_READ_ACTIONS

Objet

L'attribut POST_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés lorsque 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 l'attribut MENU doivent être exécutés lorsque tous les attributs POST_READ_ACTIONS spécifiques à l'entité sont terminés.

Structure lexicale

```
POST_READ_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
[DEFINITION { compound-statement }]<cond>?
```

Les attributs POST_READ_ACTIONS sont spécifiés dans le Tableau 82.

11.1.8 POST_WRITE_ACTIONS

Objet

L'attribut POST_WRITE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés lorsque 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 l'attribut MENU doivent être exécutés lorsque tous les attributs POST_WRITE_ACTIONS spécifiques à l'entité sont terminés.

Structure lexicale

```
POST_WRITE_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs POST_WRITE_ACTIONS sont spécifiés dans le Tableau 82.

11.1.9 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement après l'activation de l'attribut 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 l'attribut MENU doivent être exécutés avant tout attribut PRE_EDIT_ACTIONS défini des entités référencées. Lorsque les attributs PRE_EDIT_ACTIONS de l'attribut MENU sont terminés, les attributs PRE_EDIT_ACTIONS éventuels pour les entités référencées doivent être exécutés.

Structure lexicale

```
PRE_EDIT_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs PRE_EDIT_ACTIONS sont spécifiés dans le Tableau 82.

11.1.10 PRE_READ_ACTIONS

Objet

L'attribut PRE_READ_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant la lecture ou le traitement de toute entité référencée dans l'attribut ITEMS. 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. Lorsque les attributs PRE_READ_ACTIONS de l'attribut MENU sont terminés, les attributs PRE_READ_ACTIONS éventuels pour les entités référencées doivent être exécutés.

Structure lexicale

```
PRE_READ_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs PRE_READ_ACTIONS sont spécifiés dans le Tableau 82.

11.1.11 PRE_WRITE_ACTIONS

Objet

L'attribut PRE_WRITE_ACTIONS spécifie les attributs METHOD qui doivent être exécutés avant l'écriture ou le traitement de toute entité référencée dans l'attribut ITEMS. 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. Lorsque les attributs PRE_WRITE_ACTIONS de l'attribut MENU sont terminés, les attributs PRE_WRITE_ACTIONS pour les entités référencées doivent être exécutés.

Structure lexicale

```
PRE_WRITE_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs PRE_WRITE_ACTIONS sont spécifiés dans le Tableau 82.

11.1.12 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 Tableau 81) est impliqué à chaque élément sur l'attribut MENU.

Si un ou plusieurs attributs MENU sont actifs et si un attribut MENU DIALOG est activé, l'attribut MENU DIALOG est prioritaire et doit être fermé avant de pouvoir utiliser tout autre attribut MENU.

Structure lexicale

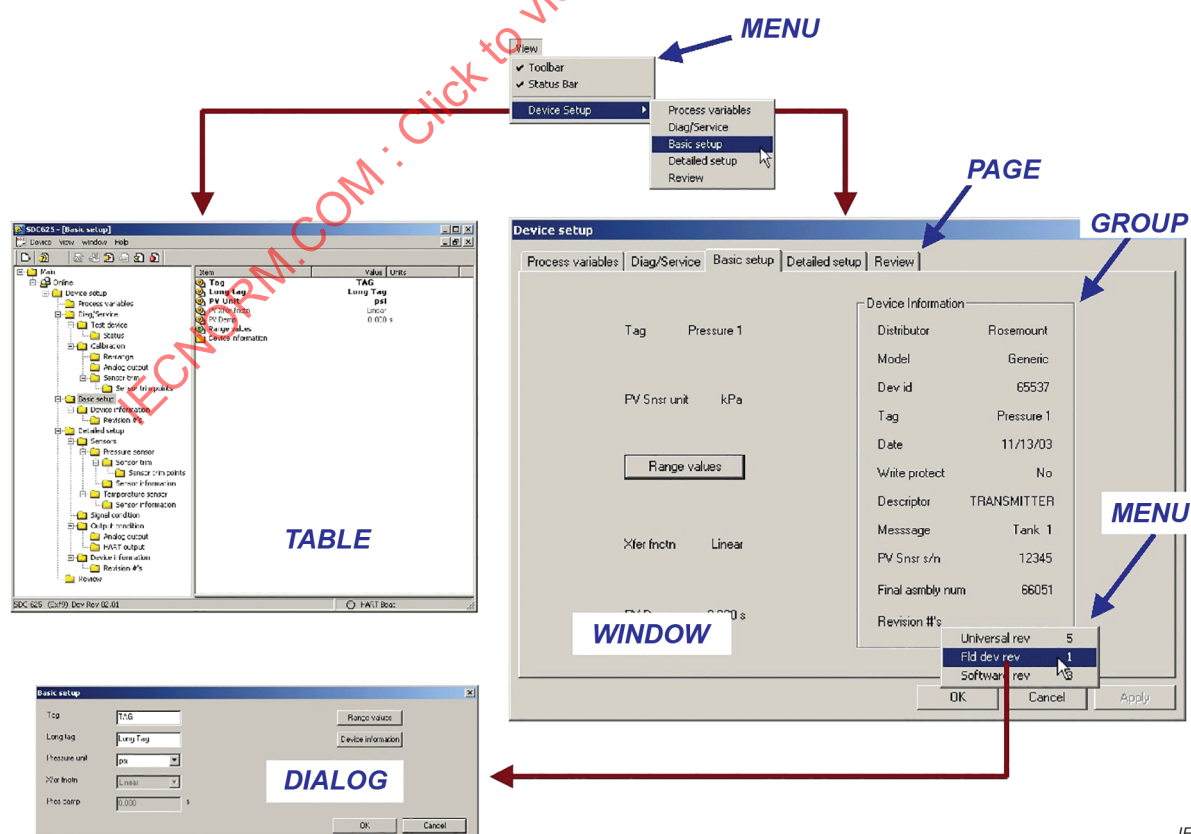
```
STYLE [DIALOG | WINDOW | PAGE | GROUP | MENU | TABLE];
```

Les attributs STYLE sont spécifiés dans le Tableau 83. Des exemples d'attributs MENU STYLE sont représentés à la Figure 41.

Tableau 83 – Attribut STYLE

Valeur d'attribut	Description
DIALOG	Aucun autre attribut MENU ne peut être activé lorsque cet attribut MENU, ou l'un de ses sous-menus, est actif. L'attribut MENU doit être fermé avant la possibilité d'interactions avec d'autres attributs MENU: il convient que les éléments de l'attribut MENU s'affichent sous forme de boîte de dialogue modale. Tous les attributs de sous-MENU doivent conserver cette caractéristique modale.
WINDOW	D'autres attributs MENU peuvent être activés pendant que cet attribut MENU est actif. Les interactions avec tous les attributs MENU activés qui ont des attributs STYLE WINDOW sont possibles: il convient que les éléments de l'attribut MENU s'affichent sous forme de fenêtre non modale.
PAGE	Il convient que les éléments de l'attribut MENU s'affichent sous forme de page de propriétés à onglets. L'attribut MENU LABEL s'affiche sur l'onglet.
GROUP	Il convient que les éléments de l'attribut MENU s'affichent sous forme de cadre de groupe. L'attribut MENU LABEL est apposé en haut du cadre de groupe. Un attribut GROUP peut inclure un attribut COLUMNBREAK et donc couvrir plusieurs colonnes.
MENU	Il convient que les éléments de l'attribut MENU s'affichent sous forme de menu déroulant ou contextuel.
TABLE	Il convient que les éléments de l'attribut MENU s'affichent en colonnes verticales. Il convient que les sous-menus soient mis en cascade et que l'utilisateur puisse naviguer dans la structure de menus résultante. Les applications EDD à affichage limité peuvent contraindre tous les menus à l'attribut STYLE TABLE.

Pour en savoir plus sur les conteneurs de menu et les éléments contenus, voir l'IEC 61804-4.



IEC

Figure 41 – Exemples d'attributs MENU STYLE

11.1.13 Schémas de séquence pour les actions

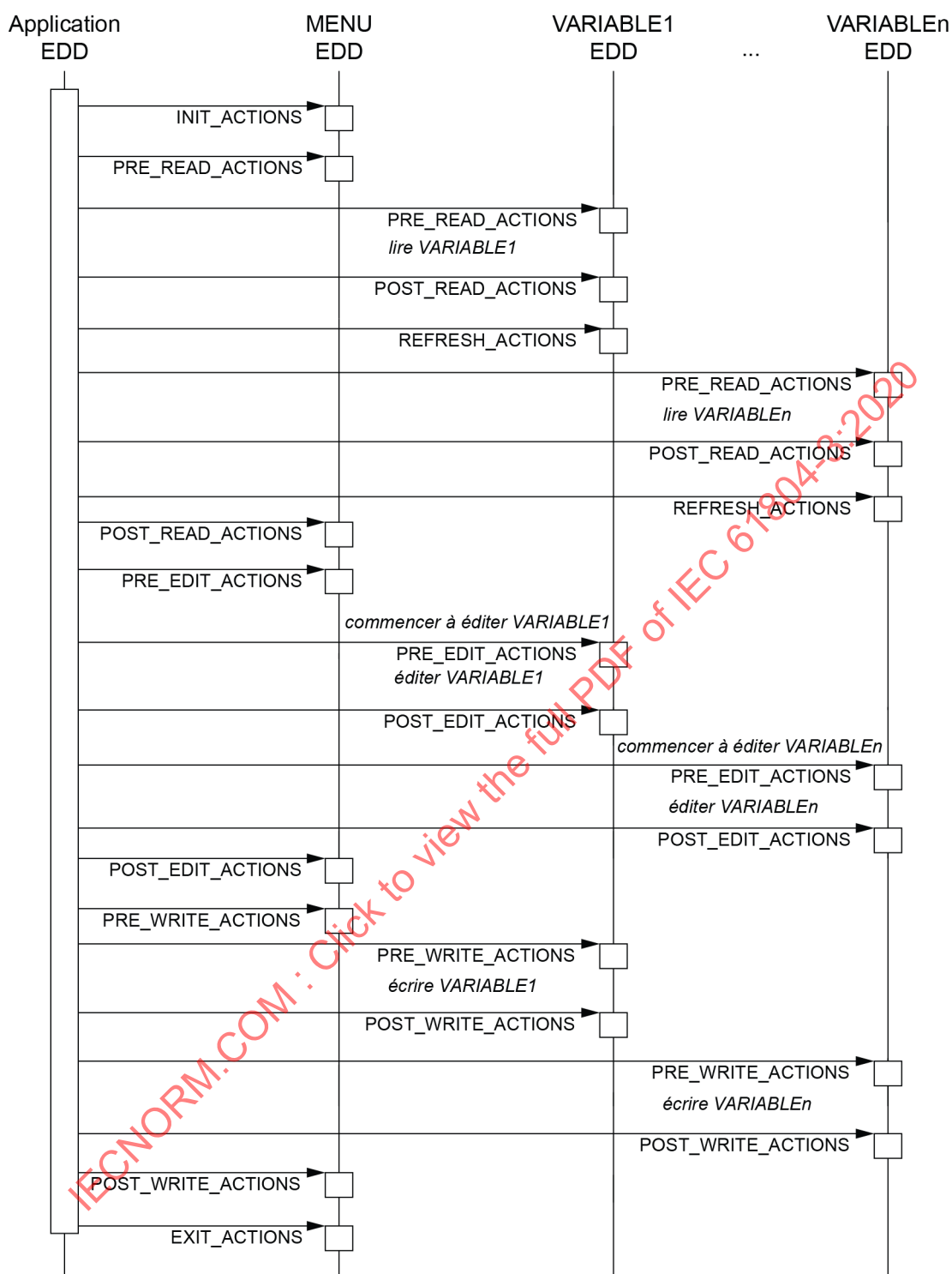
La Figure 42 représente des schémas de séquence pour des actions relatives aux attributs MENU et aux attributs VARIABLE et METHOD qu'ils contiennent. Les actions sont exécutées et achevées en séquence du haut vers le bas du schéma. Si une action échoue ou est abandonnée pour une raison imprévue, les actions restantes ne sont pas exécutées et l'attribut MENU est annulé.

Lorsque l'attribut MENU a un attribut 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 sur l'équipement), les actions pré/post-lecture/écriture pour l'attribut 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.

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2020



IEC

Figure 42 – Activation de MENU

11.2 EDIT_DISPLAY

11.2.1 Structure générale

Objet

Un attribut EDIT_DISPLAY définit la manière dont les données sont gérées pour l'affichage et l'édition. Il est utilisé pour regrouper des éléments pour édition. Les attributs EDIT_DISPLAY sont présentés de la même manière qu'une boîte de dialogue modale (un attribut MENU avec un attribut STYLE DIALOG).

Structure lexicale

EDIT_DISPLAY *identifieur*

```
{
  LABEL string-value;
  HELP string-value;
  DISPLAY_ITEMS { reference+ | reference<cond>+ }
  EDIT_ITEMS { reference+ | reference<cond>+ }
  POST_EDIT_ACTIONS { actions-specifieur }
  PRE_EDIT_ACTIONS { actions-specifieur }
  VALIDITY boolean-specifieur;
  VISIBILITY boolean-specifieur;
}
```

Les attributs EDIT_DISPLAY sont spécifiés dans le Tableau 84.

Tableau 84 – Attributs EDIT_DISPLAY

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		EDIT_DISPLAY	x	✓	x	✓	✓	x	x
11.2.3		DISPLAY_ITEMS	x	o	x	o	o	x	x
11.2.2		EDIT_ITEMS	x	m	x	m	m	x	x
14.8		HELP	x	x	x	o	x	x	x
14.9		LABEL	x	m	x	m	m	x	x
11.2.4		POST_EDIT_ACTIONS	x	o	x	o	o	x	x
11.2.5		PRE_EDIT_ACTIONS	x	o	x	o	o	x	x
14.17		VALIDITY	x	x	x	o	x	x	x
14.18		VISIBILITY	x	x	x	o	x	x	x

11.2.2 EDIT_ITEMS

Objet

L'attribut EDIT_ITEMS définit l'ensemble d'éléments de données qui doit être modifié 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 éléments spécifiques pour la présentation et l'édition d'un élément de données (par exemple, valeur par défaut, facteur d'échelle, plage) 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. Il n'est pas nécessaire que l'utilisateur modifie toutes les valeurs, mais il convient au moins qu'il valide que les valeurs actuelles sont acceptables.

Structure lexicale

EDIT_ITEMS { *reference*+ | *reference*<cond>+ }

L'attribut EDIT_ITEMS est spécifié dans le Tableau 85.

Tableau 85 – Attribut EDIT_ITEMS

Type de valeurs	Description
reference	Profils de communication: HART Référence à une instance de l'attribut VARIABLE ou WRITE_AS_ONE.
reference<cond>	Profils de communication: FF, ISA100 Référence à une instance de l'attribut VARIABLE, WRITE_AS_ONE, BLOCK_A PARAMETERS ou à un élément de l'attribut BLOCK_A PARAMETERS. Peut-être défini par une condition.

11.2.3 DISPLAY_ITEMS

Objet

L'attribut DISPLAY_ITEMS définit l'ensemble d'éléments de données qui doit être présenté à l'utilisateur pour information; ces éléments ne doivent pas être modifiés: 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 éléments spécifiques pour la présentation d'un élément de données (par exemple, valeur par défaut, facteur d'échelle, plage) sont spécifiés dans les définitions de chaque élément individuel référencé.

Structure lexicale

DISPLAY_ITEMS { *reference*+ | *reference*<cond>+ }

L'attribut DISPLAY_ITEMS est spécifié dans le Tableau 86.

Tableau 86 – Attributs DISPLAY_ITEM

Type de valeurs	Description
reference	Profils de communication: HART Référence à un attribut VARIABLE.
reference<cond>	Profils de communication: FF, ISA100 Référence à une instance de l'attribut VARIABLE, WRITE_AS_ONE, BLOCK_A PARAMETERS ou à un élément de l'attribut BLOCK_A PARAMETERS. Peut-être défini par une condition.

11.2.4 POST_EDIT_ACTIONS

Objet

L'attribut POST_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés lorsque l'utilisateur a fini le traitement de l'attribut EDIT_DISPLAY. Si l'attribut 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 est entrée. Les attributs POST_EDIT_ACTIONS de l'attribut EDIT_DISPLAY sont exécutés uniquement lorsque tous ces attributs POST_EDIT_ACTIONS individuels sont terminés.

Structure lexicale

```
POST_EDIT_ACTIONS
{
    reference<cond> [, reference<cond>]*
}
```

Les attributs POST_EDIT_ACTIONS sont spécifiés dans le Tableau 87.

11.2.5 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS spécifie les attributs METHOD qui doivent être exécutés immédiatement lorsque l'attribut 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 de l'attribut EDIT_DISPLAY doivent être exécutés avant tout attribut PRE_EDIT_ACTIONS défini des entités référencées.

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 n'active pas les attributs PRE_EDIT_ACTIONS des objets VARIABLE.

Structure lexicale

```
PRE_EDIT_ACTIONS
{
    reference<cond> [, reference<cond>]*
}
```

Les attributs PRE_EDIT_ACTIONS sont spécifiés dans le Tableau 87.

Tableau 87 – Attributs POST_EDIT_ACTIONS et PRE_EDIT_ACTIONS

Attribut	Description
reference	Référence à une instance de METHOD ou à une instance de METHOD avec arguments.

11.3 IMAGE

11.3.1 Structure générale

Objet

La construction de base IMAGE est utilisée pour afficher des images sur l'interface utilisateur.

Structure lexicale

IMAGE *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  LINK reference<cond>;
  PATH string-literal<cond>;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}
```

Les attributs IMAGE sont spécifiés dans le Tableau 88.

Tableau 88 – Attributs IMAGE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	IMAGE		✓	✓	✓	✓	✓	✓	✓
14.8	HELP		o	o	o	o	o	o	o
14.9	LABEL		o	o	o	o	o	o	o
11.3.2	LINK		o	o	o	o	o	o	o
11.3.3	PATH		m	m	m	m	m	m	m
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	o	o	o	o

Utilisation

La Figure 43 représente une définition d'image.

```
IMAGE Sensor_Diagram
{
  LABEL "Sensor Diagram";
  HELP "This Diagram shows the sensor characteristic";
  PATH "SenDiaEn.jpg\
    |en zz|SenDiaEn_LowRes.jpg\
    |de|SenDiaDe.jpg\
    |de zz|SenDiaDe_LowRes.jpg";
}
```

Figure 43 – Exemple d'EDD d'un attribut IMAGE

Une définition IMAGE peut être utilisée pour invoquer un attribut METHOD ou un attribut MENU de STYLE DIALOG ou WINDOW avec l'attribut LINK.

La Figure 44 représente un exemple d'EDD pour l'utilisation de l'attribut LINK dans une image.

```

MENU
{
    ITEMS
    {
        ...,
        Self_Test_Image (INLINE),
        ...
    }
}

IMAGE Self_Test_Image
{
    LABEL "Self Test";
    HELP "This Method execute the self test function of the device which needs some time";
    PATH "SelfTestImage.jpg";
    LINK Self_Test_Menu;
}

MENU Self_Test_Menu
{
    ...
}
    
```

Figure 44 – Exemple d'EDD d'un attribut IMAGE avec l'attribut LINK

11.3.2 LINK

Objet

L'attribut LINK spécifie un élément EDD qui peut être accessible depuis l'attribut IMAGE. Cela permet d'obtenir des informations supplémentaires concernant l'attribut IMAGE. Lorsque l'attribut IMAGE est affiché, la sélection de l'image entraîne l'activation de l'élément EDD objet du lien hypertexte.

Un attribut MENU référencé doit être de style DIALOG ou WINDOW. Dans les deux cas, il doit être restitué selon le style DIALOG.

Structure lexicale

LINK *reference*<cond>;

L'attribut LINK est spécifié dans le Tableau 89.

Tableau 89 – Attribut LINK

Type de valeurs	Description
reference<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Référence à une instance de l'attribut MENU, METHOD ou PLUGIN; peut-être définie par une condition
reference<cond>	Profils de communication: HART Référence à une instance de l'attribut MENU, METHOD, PLUGIN ou EDIT_DISPLAY.

11.3.3 PATH

Objet

L'attribut PATH spécifie le nom de fichier de l'image. Des images pour des paramètres régionaux multiples peuvent être spécifiées en utilisant la même technique de localisation que celle utilisée avec les littéraux de chaîne. Le code de pays zz peut être utilisé pour spécifier une version basse résolution de l'image, ce qui est utile pour les applications EDD qui fonctionnent sur des équipements 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*<cond>;

L'attribut PATH est spécifié dans le Tableau 90.

Tableau 90 – Attribut PATH

Type de valeurs	Description
file-name<cond>	Littéral de chaîne qui identifie le nom de fichier de l'image; peut-être défini par une condition. Le Tableau 91 spécifie les formats d'image admis.

Les formats d'images du Tableau 91 sont admis pour une utilisation avec un attribut IMAGE.

Tableau 91 – Formats d'image

Format	Description
JPG, JPEG	Format conforme à l'ISO/IEC 10918-1
PNG	Format Portable Network Graphics conforme à l'ISO/IEC 15948
GIF	Graphics Interchange Format NOTE Certaines applications EDD peuvent ne pas prendre en charge les animations GIF.

11.4 GRID

11.4.1 Structure générale

Objet

L'attribut GRID décrit une matrice de données qui doivent 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
{
  LABEL string-value;
  HELP string-value;
  HANDLING handling-specifier;
  HEIGHT height-width-specifier;
  ORIENTATION orientation-specifier;
  VALIDITY boolean-specifier;
  VECTORS { vectors-specifier }
  VISIBILITY boolean-specifier;
  WIDTH height-width-specifier;
}

```

Les attributs GRID sont spécifiés dans le Tableau 92.

Tableau 92 – Attributs GRID

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	GRID		✓	✓	✓	✓	✓	✓	✓
14.6		HANDLING	o	o	o	o	o	o	o
14.7		HEIGHT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
11.4.2		ORIENTATION	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
11.4.3		VECTORS	m	m	m	m	m	m	m
14.18		VISIBILITY	o	o	o	o	o	o	o
14.19		WIDTH	o	o	o	o	o	o	o

Utilisation

La Figure 45 représente un exemple d'attribut GRID.

```

VARIABLE PeakType
{
    LABEL "Peak Type";
    CLASS DEVICE;
    TYPE ENUMERATED
    {
        { 1, "False Echo"},
        { 2, "Button Echo" },
        { 3, "Unknown" }
    }
}

ARRAY arrPeakType
{
    NUMBER_OF_ELEMENTS 10;
    TYPE PeakType;
}

VARIABLE PeakDistance
{
    LABEL "Peak Distance";
    CLASS DEVICE;
    TYPE FLOAT
    {
        DEFAULT_VALUE 1.0;
    }
}

ARRAY arrPeakDistance
{
    LABEL "Peak Distance";
    NUMBER_OF_ELEMENTS 10;
    TYPE PeakDistance;
}

VARIABLE PeakAmplitude
{
    LABEL "Device Amplitude";
    CLASS DEVICE;
    TYPE FLOAT
    {
        DEFAULT_VALUE 1.0;
    }
}

ARRAY arrPeakAmplitude
{

```



```

        LABEL "Peak Amplitudes";
        NUMBER_OF_ELEMENTS 10;
        TYPE PeakAmplitude;
    }

MENU DeviceEcho
{
    LABEL "Device Echo";
    ITEMS
    {
        EchoCurve,
        FoundEcho,
        FalseEchos
    }
}

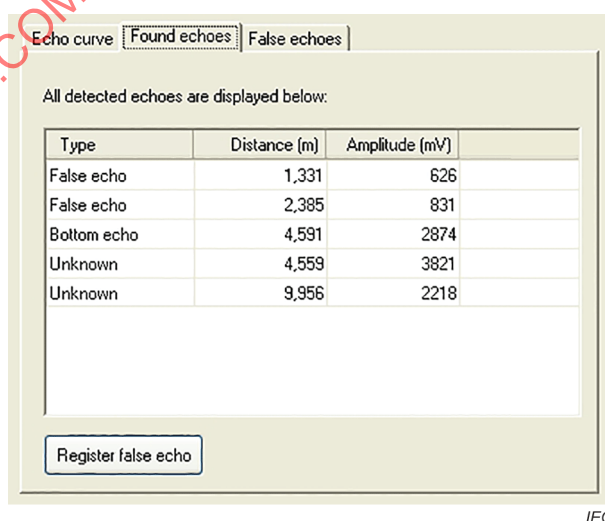
MENU FoundEcho
{
    LABEL "Found echoes";
    STYLE PAGE;
    ITEMS
    {
        GridFoundEcho,
        RegisterFalseEchoes
    }
}

GRID GridFoundEchoes
{
    LABEL "All detected echoes are displayed below";
    VALIDITY IF (varModelCode == 4711) { TRUE; } ELSE { FALSE; }
    VECTORS
    {
        {"Type", arrPeakType},
        {"Distance (m)", arrPeakDistance},
        {"Amplitude (mV)", arrPeakAmplitude}
    }
}

```

Figure 45 – Exemple d'EDD d'un attribut GRID

La Figure 46 représente les résultats de l'exemple d'EDD ci-dessus. Les unités constantes ou les unités n'apparaissent pas avec les valeurs lorsqu'elles sont affichées dans l'attribut GRID. Sur la Figure 46, l'indication d'unité visible dans l'en-tête de la colonne (par exemple "(m)") fait partie de la déclaration VECTORS.



Type	Distance (m)	Amplitude (mV)
False echo	1,331	626
False echo	2,385	831
Bottom echo	4,591	2874
Unknown	4,559	3821
Unknown	9,956	2218

Register false echo

IEC

Figure 46 – Résultats de l'exemple d'EDD

11.4.2 ORIENTATION

Objet

L'attribut ORIENTATION spécifie la direction dans laquelle les vecteurs sont affichés. Lorsqu'aucun attribut ORIENTATION n'est spécifié, il est défini par défaut sur VERTICAL et les attributs VECTOR s'affichent verticalement.

Structure lexicale

```
ORIENTATION orientation-spezifier;  
orientation-spezifier:= [[HORIZONTAL | VERTICAL] | [HORIZONTAL |  
VERTICAL]<cond>]
```

Les attributs ORIENTATION sont spécifiés dans le Tableau 93.

Tableau 93 – Attribut ORIENTATION

Valeur d'attribut	Description
[HORIZONTAL VERTICAL]<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Les vecteurs s'affichent en lignes (HORIZONTAL) ou en colonnes (VERTICAL); ils peuvent être définis par une condition
[HORIZONTAL VERTICAL]	Profils de communication: HART Les vecteurs s'affichent en lignes (HORIZONTAL) ou en colonnes (VERTICAL)

11.4.3 VECTORS

Objet

L'attribut VECTORS spécifie le contenu de l'attribut GRID. Les attributs VECTOR s'affichent dans leur ordre d'apparition. 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 fournit un libellé pour le vecteur suivi par les données à afficher. Même si elles sont obligatoires, les descriptions peuvent être une chaîne vide, auquel cas l'en-tête est vierge. Si toutes les descriptions sont des chaînes vides, aucune cellule d'en-tête n'est affichée.

Structure lexicale

```
VECTORS { vectors-spezifier }  
vectors-spezifier:=  
{  
  [vector-spezifier ,]<cond>+  
}  
vector-spezifier:=  
[description [, [string-constant | double-constant | float-constant |  
integer-constant | unsigned-integer-constant]]+] |  
[description [, item-reference]+] |  
[description , array-reference ]
```

Les attributs VECTORS sont spécifiés dans le Tableau 94.

Tableau 94 – Attribut VECTORS

Utilisation	Attribut	Description
m	description	Chaîne contenant une brève description des données contenues dans le vecteur.
o	item-reference	Référence à une instance de l'attribut VARIABLE, à une instance de l'attribut IMAGE, à un membre d'une instance de l'attribut RECORD, à un élément d'une instance de l'attribut VALUE_ARRAY ou à un élément d'une instance de l'attribut REFERENCE_ARRAY.
o	array-reference	Référence à une instance de l'attribut VALUE_ARRAY ou à une instance de l'attribut REFERENCE_ARRAY.
o	string-constant	Chaîne constante ou référence représentée par une chaîne.
o	double-constant	Constante à virgule flottante à double précision.
o	float-constant	Constante à virgule flottante à simple précision.
o	integer-constant	Constante entière.
o	unsigned-integer-constant	Constante entière non signée.

11.5 CHART

11.5.1 Structure générale

Objet

L'attribut CHART décrit un diagramme utilisé pour afficher des données provenant d'un équipement dans l'application EDD, celles-ci variant dans le temps. Un attribut CHART est utilisé pour afficher des valeurs de données continues provenant de l'équipement, par opposition à un attribut GRAPH qui est utilisé pour afficher un ensemble de données statiques. Un ou plusieurs attributs SOURCE sont spécifiés pour l'attribut CHART que l'application EDD doit afficher. Les données provenant de l'équipement peuvent être ajustées avant d'être affichées par l'utilisation d'attributs INIT_ACTIONS et REFRESH_ACTIONS sur les membres de l'attribut SOURCE de l'attribut CHART.

La couleur d'arrière-plan d'un affichage CHART est déterminée par l'application EDD. Afin de garantir une expérience commune, il est recommandé de choisir une couleur d'arrière-plan blanche. Il est recommandé qu'une application EDD prenne en charge l'affichage simultané d'au moins six courbes.

L'EDD peut fournir des informations relatives aux libellés et aux aides pour les attributs CHART, leurs courbes et leurs axes. Pour établir les légendes et les informations d'aide pour les courbes, l'application EDD doit prendre en charge les règles définies en 11.5.6 ci-dessous.

Il convient que les applications EDDL ne prenant pas en charge la vue graphique de l'attribut CHART affichent les variables correspondantes d'une manière numérique normalisée. Par exemple, les données peuvent être affichées dans un format tabulaire. Il convient que le format soit choisi par le développeur de l'application EDD.

Structure lexicale

CHART *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  CYCLE_TIME integer-expression | integer-expression<cond>;
  HEIGHT height-width-specifier;
  LENGTH integer-expression | integer-expression<cond>;
  MEMBERS { members-specifier }
  TYPE chart-type-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  WIDTH height-width-specifier;
}
```

Les attributs CHART sont spécifiés dans le Tableau 95.

Tableau 95 – Attributs CHART

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		CHART	✓	✓	✓	✓	✓	✓	✓
11.5.2		CYCLE_TIME	o	o	o	o	o	o	o
14.7		HEIGHT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
11.5.3		LENGTH	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
11.5.4		TYPE	o	o	o	o	o	o	o
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o
14.19		WIDTH	o	o	o	o	o	o	o

Utilisation

La Figure 47 représente un diagramme avec une courbe dans une boîte de dialogue. La courbe est actualisée selon une durée de cycle de 1 s.

```

MENU measuring_values
{
    LABEL "Measuring Values";
    STYLE DIALOG;
    ITEMS
    {
        primary_value_view
    }
}

MENU primary_value_view
{
    LABEL "Primary Measuring Value";
    STYLE PAGE;
    ITEMS
    {
        primary_value_chart
    }
}

VARIABLE primary_value
{
    LABEL "Primary Value";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "Bar";
}

SOURCE primary_value_source
{
    MEMBERS
    {
        PRIM_VAL, primary_value;
    }
}

CHART primary_value_chart
{
    LABEL "Primary Value";
    MEMBERS
    {
        CHART1, primary_value_source;
    }
}

```

Figure 47 – Exemple de diagramme avec une courbe dans une boîte de dialogue

La Figure 48 est un exemple d'EDD qui représente la combinaison de deux attributs SOURCE. La Figure 49 représente comment une application EDD peut afficher le diagramme.

```

VARIABLE primary_value
{
    LABEL "Primary Value";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

VARIABLE primary_value_undamped
{
    LABEL "Undamped PV";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

AXIS axis_0_14
{
    LABEL "axis 0-14";
    MIN_VALUE 0;
    MAX_VALUE 14;
}

SOURCE primary_value_stack1
{
    MEMBERS
    {
        PRIM_VAL_1, primary_value;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0x0000FF; /*BLEU*/
}

SOURCE primary_value_stack2
{
    MEMBERS
    {
        PRIM_VAL_2, primary_value_undamped;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0xFF0000; /*ROUGE*/
}

CHART primary_value_stack_chart
{
    LABEL "Primary Value";
    MEMBERS
    {
        CHART1, primary_value_stack1, "ph1";
        CHART2, primary_value_stack2, "pH2";
    }
    TYPE VERTICAL_BAR;
}

```

Figure 48 – Exemple de diagramme avec deux attributs SOURCE

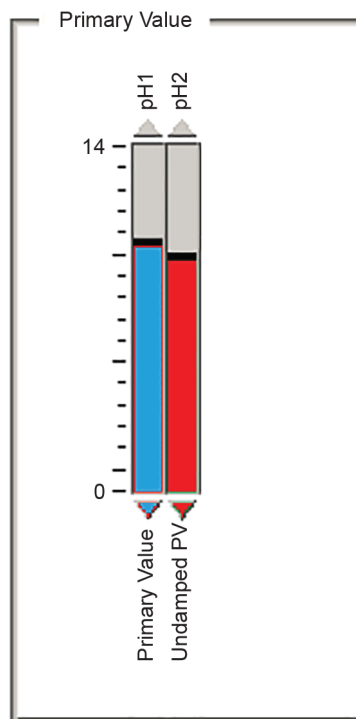


Figure 49 – Affichage d'un exemple de diagramme avec deux attributs SOURCE

La Figure 50 est un exemple d'EDD qui représente la combinaison de trois attributs SOURCE à l'horizontale. La Figure 51 représente comment une application EDD peut afficher le diagramme.

```
VARIABLE primary_value
{
    LABEL "Primary Value";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

VARIABLE primary_value_undamped
{
    LABEL "Undamped PV";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

VARIABLE primary_value_non_temperature_compensated
{
    LABEL "PV non-TC";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT "pH";
}

AXIS axis_0_14
{
    LABEL "axis 0-14";
    MIN_VALUE 0;
    MAX_VALUE 14;
}

AXIS axis_0_7
{
    LABEL "axis 0-7";
    MIN_VALUE 0;
    MAX_VALUE 7;
}
```

```

SOURCE primary_value_stack1
{
    MEMBERS
    {
        PRIM_VAL_1, primary_value;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0x0000FF; /*BLEU*/
}

SOURCE primary_value_stack2
{
    MEMBERS
    {
        PRIM_VAL_2, primary_value_undamped;
    }
    Y_AXIS axis_0_14;
    LINE_COLOR 0xFF0000; /*ROUGE*/
}

SOURCE primary_value_single
{
    MEMBERS
    {
        PRIM_VAL_3, primary_value_non_temperature_compensated;
    }
    Y_AXIS axis_0_7;
    LINE_COLOR 0x008000; /*VERT*/
}

CHART primary_value_stack_chart
{
    LABEL "Primary Value";
    MEMBERS
    {
        CHART1, primary_value_stack1, "pH1";
        CHART2, primary_value_single, "pH non-TC";
        CHART3, primary_value_stack2, "pH2";
    }
    TYPE HORIZONTAL_BAR;
}

```

Figure 50 – Exemple de diagramme avec trois barres horizontales

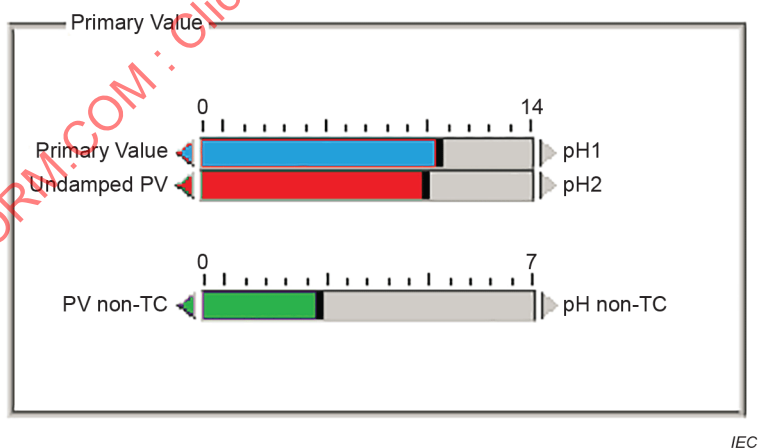


Figure 51 – Affichage d'un exemple de diagramme avec trois barres horizontales

Les attributs facultatifs INIT_ACTIONS et/ou REFRESH_ACTIONS peuvent être définis dans un attribut SOURCE.

La même méthode peut être référencée par les listes de méthodes INIT_ACTIONS et REFRESH_ACTIONS.

La Figure 52 représente un exemple de l'utilisation des méthodes dans un diagramme. Cet exemple représente un diagramme dans une boîte de dialogue.


```

MENU measuring_values
{
    LABEL "Measuring Values";
    STYLE DIALOG;
    ITEMS
    {
        primary_value_view
    }
}

MENU primary_value_view
{
    LABEL "Primary Measuring Value";
    STYLE PAGE;
    ITEMS
    {
        primary_value_chart
    }
}

VARIABLE primary_value
{
    LABEL "Primary Value";
    CLASS DYNAMIC;
    TYPE FLOAT;
}

SOURCE primary_value_source
{
    LABEL "Primary";                                     // ce libellé est utilisé dans la légende
    LINE TYPE DATA1;
    EMPHASIS TRUE;
    Y_AXIS measuring_values_axis;
    MEMBERS
    {
        PRIM_VAL, primary_value;
    }
}

METHOD CalculationMethod
{
    LABEL "";
    DEFINITION
    {
        calculated_value = primary_value * 0.12345;
    }
}

AXIS measuring_values_axis
{
    LABEL "measurement value";
    MIN_VALUE 0;
    MAX_VALUE 100;
}

VARIABLE primary_value_unit
{
    LABEL "Primary Value Unit";
    CLASS CONTAINED;
    TYPE ENUMERATED(1)
    {
        { 32, [degC], [degC_help] },
        { 33, [degF], [degF_help] },
        { 35, [Kelvin], [Kelvin_help] }
    }
}

UNIT
{
    primary_value_unit:
    primary_value,
    calculated_value,
    measuring_values_axis
}

SOURCE calculated_value_source
{
    LABEL "calculated";                                   // ce libellé est utilisé dans la légende

```

```

    LINE_TYPE DATA2;
    Y_AXIS measuring_values_axis;
    MEMBERS
    {
        CALC_VAL, calculated_value;
    }
    INIT_ACTIONS    {CalculationMethod}
    REFRESH_ACTIONS {CalculationMethod}
}

CHART primary_value_chart
{
    LABEL "Primary Value";
    LENGTH 600000;
    TYPE SCOPE;
    WIDTH LARGE;
    HEIGHT SMALL;
    CYCLE_TIME 5000; /*Valeur source actualisée toutes les 5 000 msec*/
    MEMBERS
    {
        GRAPH1, primary_value_source;
        GRAPH2, calculated_value_source;
    }
}

```

Figure 52 – Exemple de diagramme dans une boîte de dialogue

11.5.2 CYCLE_TIME

Objet

L'attribut **CYCLE_TIME** spécifie la durée, en millisecondes (ms), pendant laquelle l'application EDD doit attendre entre des lectures successives des valeurs sources (et de l'attribut **REFRESH_ACTIONS** invoqué) de l'équipement. Par défaut, la durée de cycle d'un attribut **CHART** sans attribut **CYCLE_TIME** est de 1 000 ms.

NOTE Pour des raisons de performance de l'application EDD (performances moindres d'une E/S multiplexée, par exemple), la fréquence de mise à jour, les latences et les giges peuvent entraîner une variation des durées de cycle réelles. Si l'application EDD ne peut pas lire à la cadence définie par la durée de cycle, elle lit simplement les données le plus rapidement possible.

Même si l'attribut **CYCLE_TIME** peut faire référence à un attribut **VARIABLE**, l'application EDD doit conserver un attribut **CYCLE_TIME** fixe et invariable aussi longtemps que l'attribut **CHART** est affiché. L'attribut **CYCLE_TIME** doit être résolu et capturé immédiatement après l'exécution de l'attribut **INIT_ACTIONS** et avant l'affichage de l'attribut **CHART**.

L'attribut **CYCLE_TIME** doit être fixe lorsque l'attribut **CHART** est affiché. Si l'attribut **CHART** est fermé, puis rouvert, il peut obtenir un nouvel attribut **CYCLE_TIME**.

Profils de communication: HART

La limite inférieure de l'attribut **CYCLE_TIME** est de 1 000 ms. Les attributs **CYCLE_TIME** plus rapides doivent entraîner un attribut **CYCLE_TIME** de 1 000 ms.

Structure lexicale

CYCLE_TIME *integer-expression* | *integer-expression*<cond>;

L'attribut **CYCLE_TIME** est spécifié dans le Tableau 96.

Tableau 96 – Attribut CYCLE_TIME

Type de valeurs	Description
integer-expression	Profils de communication: HART Temps, en millisecondes (ms), entre des lectures successives des valeurs sources de l'équipement.
integer-expression<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Temps, en millisecondes (ms), entre des lectures successives des valeurs sources de l'équipement. Peut-être défini par une condition.

11.5.3 LENGTH

Objet

L'attribut LENGTH spécifie la durée, en millisecondes (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. Même si l'attribut LENGTH peut faire référence à un attribut VARIABLE, l'application EDD doit conserver un attribut LENGTH fixe et invariable aussi longtemps que l'attribut CHART est affiché. Si l'attribut CHART est fermé, puis rouvert, il peut obtenir un nouvel attribut LENGTH.

Structure lexicale

LENGTH *integer-expression* | *integer-expression<cond>*;

L'attribut LENGTH est spécifié dans le Tableau 97.

Tableau 97 – Attribut LENGTH

Attribut	Description
integer-expression	Profils de communication: HART Temps, en millisecondes (ms), qui est affiché par l'attribut CHART. L'attribut LENGTH doit être résolu par une valeur positive supérieure ou égale à 60 secondes.
integer-expression<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Temps, en millisecondes (ms), qui est affiché par l'attribut CHART. Peut-être défini par une condition.

11.5.4 TYPE

Objet

L'attribut TYPE spécifie le type de CHART que l'EDD doit afficher. Trois (3) types d'instruments de mesure et trois (3) types de tendances sont pris en charge. Les types d'instruments de mesure sont GAUGE, HORIZONTAL_BAR et VERTICAL_BAR. Ceux-ci sont représentés à la Figure 54. Les diagrammes de tendances sont: SCOPE, STRIP et SWEEP. Ceux-ci sont représentés à la Figure 53. Par défaut, le type d'un attribut CHART sans attribut TYPE est STRIP.

Pour les diagrammes de types d'instruments de mesure, les attributs SOURCE avec un même attribut Y_AXIS doivent être regroupés et affichés avec un instrument de mesure unique. L'instrument de mesure doit être labellisé à l'aide de l'attribut LABEL de l'attribut Y_AXIS. Chaque attribut Y_AXIS spécifié par l'attribut CHART se traduit par un autre instrument de mesure et tous les instruments de mesure de l'attribut CHART sont empilés ensemble. La couleur de l'indication de la variable est déterminée à l'aide des règles données en 14.10. Les instruments de mesure sont regroupés avec le libellé de l'attribut CHART à des fins de référence utilisateur.

Structure lexicale

TYPE *chart-type-specifier*;

Profils de communication: CS, FF, GPE, ISA100, PB, PN

```
chart-type-specifier:= [GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |
SWEEP | VERTICAL_BAR] | [[GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |
SWEEP | VERTICAL_BAR]<cond>]
```

Profils de communication: HART

```
chart-type-specifier:= [GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |
SWEEP | VERTICAL_BAR] | [[GAUGE | HORIZONTAL_BAR | SCOPE | STRIP |
SWEEP | VERTICAL_BAR]
```

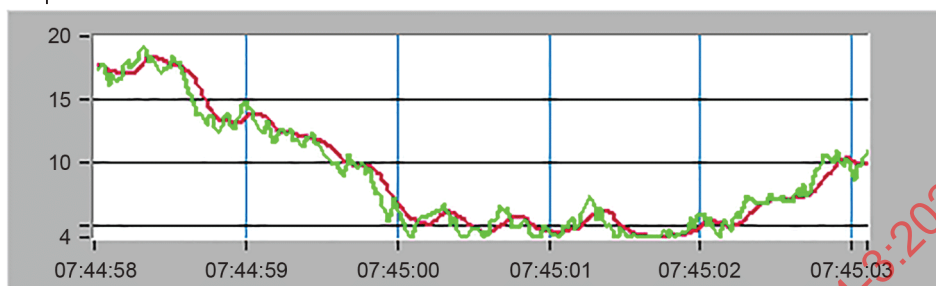
Les attributs TYPE sont spécifiés dans le Tableau 98.

Tableau 98 – Attribut TYPE

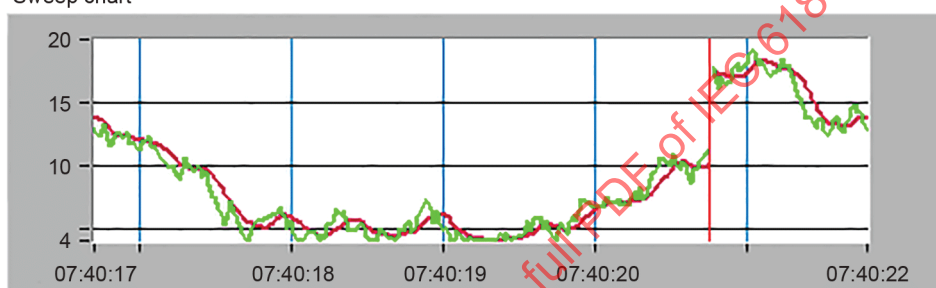
Valeur d'attribut	Description
GAUGE	Une valeur source unique s'affiche sous la forme d'une jauge, laquelle est un instrument de mesure analogique circulaire à cadran. L'indicateur est une flèche ou un pointeur qui tourne en fonction de la variation de la valeur source actuelle. Les valeurs minimales et maximales se trouvent généralement en bas du cadran et l'indicateur tourne dans le sens horaire lorsque la valeur source augmente. Un attribut GAUGE est une représentation graphique des données similaire à la jauge de carburant d'une automobile.
HORIZONTAL_BAR	Les valeurs sources s'affichent sous la forme d'un diagramme à barres horizontales dont la longueur est proportionnelle à la valeur source actuelle. La barre s'affiche sur le plan horizontal, l'origine (valeur minimale) étant située sur la gauche du diagramme.
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 l'attribut CHART, la zone d'affichage est effacée et les nouvelles valeurs sources sont à nouveau affichées à partir de l'extrémité gauche de la zone d'affichage. L'axe X doit être mis à jour correctement en fonction de la zone d'affichage.
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 l'attribut CHART, la zone d'affichage est décalée, les nouvelles données s'ajoutent à l'extrémité droite de l'attribut CHART et les données historiques sont décalées vers la gauche. Les valeurs sources les plus anciennes sont décalées à l'extrémité gauche de l'attribut CHART puis retirées de la zone d'affichage. L'axe X doit être parcouru et mis à jour correctement en fonction de la zone d'affichage.
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 l'attribut 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.

Valeur d'attribut	Description
VERTICAL_BAR	Les valeurs sources s'affichent sous la forme d'un diagramme à barres verticales dont la longueur est proportionnelle à la valeur source actuelle. La barre s'affiche verticalement, l'origine (valeur minimale) étant située en bas du diagramme.

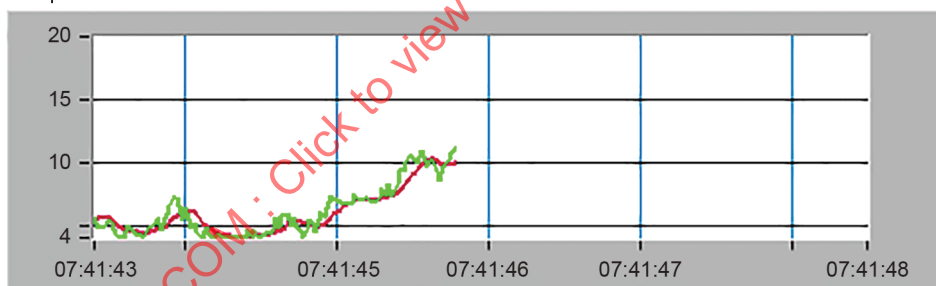
Strip chart



Sweep chart

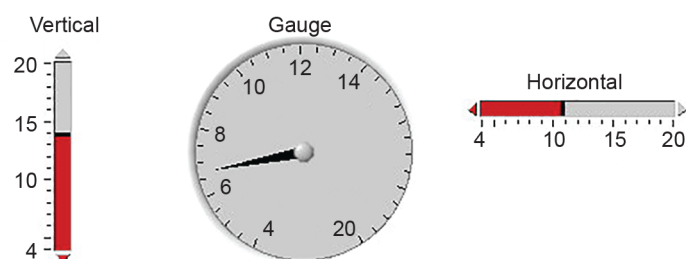


Scope chart



IEC

Figure 53 – Exemple de types d'attributs CHART de tendances



IEC

Figure 54 – Exemple de types d'attributs CHART d'instruments de mesure

L'attribut LABEL des attributs SOURCE et VARIABLE doit être utilisé pour annoter le diagramme. Ces annotations peuvent intervenir directement sur l'instrument de mesure où apparaître dans une légende. Le libellé/l'annotation de chaque indicateur est généré à l'aide des règles exposées dans l'IEC 61804-4.

11.5.5 Schéma de séquence pour les actions

La Figure 55 représente le schéma de séquence pour des actions relatives aux attributs CHART et aux membres de l'attribut SOURCE qu'ils contiennent. Les actions sont exécutées et achevées en séquence du haut vers le bas du schéma.

Les données provenant de l'équipement peuvent être ajustées avant d'être affichées par l'utilisation d'attributs INIT_ACTIONS et REFRESH_ACTIONS sur les membres de l'attribut SOURCE de l'attribut CHART. Tous les attributs INIT_ACTIONS devant être exécutés avant l'affichage de l'attribut CHART, les attributs CYCLE_TIME et LENGTH peuvent être ajustés dans l'attribut INIT_ACTIONS sur les membres de l'attribut SOURCE.

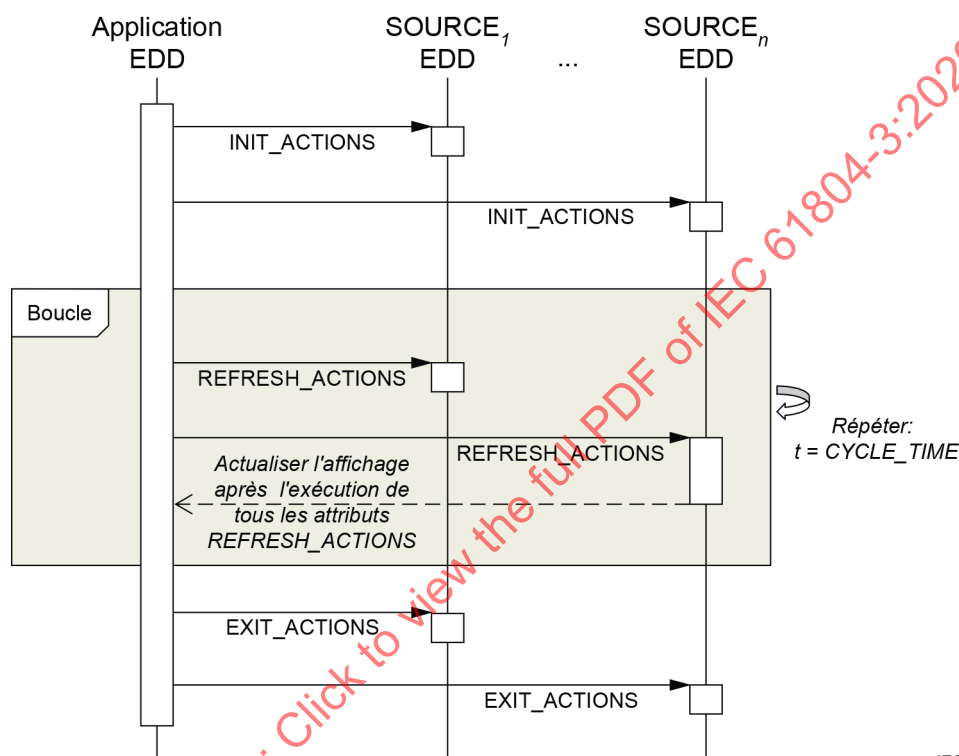


Figure 55 – Activation de l'attribut CHART

11.5.6 Règles relatives aux légendes/aides de l'attribut CHART

11.5.6.1 Instruments de mesure et tendances (attribut Y_AXIS non défini) – Un attribut VARIABLE par attribut SOURCE

- L'application EDD doit afficher le libellé de la légende/l'aide avec la description/l'aide du membre de l'attribut CHART.
- Si la description/l'aide du membre de l'attribut CHART n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec le libellé/l'aide de la référence source.
- Si le libellé/l'aide de l'attribut SOURCE n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec la description/l'aide du membre de l'attribut SOURCE.
- Si la description/l'aide du membre de l'attribut SOURCE n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec le libellé/l'aide de la référence de l'attribut VARIABLE (pour le libellé et l'aide d'une référence de variables, voir également Concaténation des libellés pour les références de variables indirectes et Concaténation des aides dans l'IEC 61804-4).

11.5.6.2 Tendances (attribut Y_AXIS non défini) – Deux attributs VARIABLE ou plus par attribut SOURCE

- L'application EDD doit afficher le libellé de la légende/l'aide d'un diagramme de tendances (STRIP, SWEEP et SCOPE) avec la description/l'aide du membre de l'attribut SOURCE.
- Si la description/l'aide du membre de l'attribut SOURCE n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec le libellé/l'aide de la référence de l'attribut VARIABLE (pour le libellé et l'aide d'une référence de variables, voir également Concaténation des libellés pour les références de variables indirectes et Concaténation des aides dans l'IEC 61804-4).

11.5.6.3 Instruments de mesure (attribut Y_AXIS non défini) – Deux attributs VARIABLE ou plus par attribut SOURCE

- L'application EDD doit afficher la légende (des différentes sources) d'un diagramme d'instruments de mesure (HORIZONTAL_BAR, VERTICAL_BAR et GAUGE) avec la description/l'aide du membre de l'attribut CHART.
- Si la description/l'aide du membre de l'attribut CHART n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec le libellé/l'aide de la référence source.
- En outre, le diagramme doit afficher un libellé/une aide de "légende à barres" supplémentaire pour chaque barre avec la description/l'aide du membre de l'attribut SOURCE.
- Si la description/l'aide du membre de l'attribut SOURCE n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec le libellé/l'aide de la référence de l'attribut VARIABLE (pour le libellé et l'aide d'une référence de variables, voir également Concaténation des libellés pour les références de variables indirectes et Concaténation des aides dans l'IEC 61804-4).

Se référer à la Figure 56, qui représente des exemples de légendes de diagrammes.

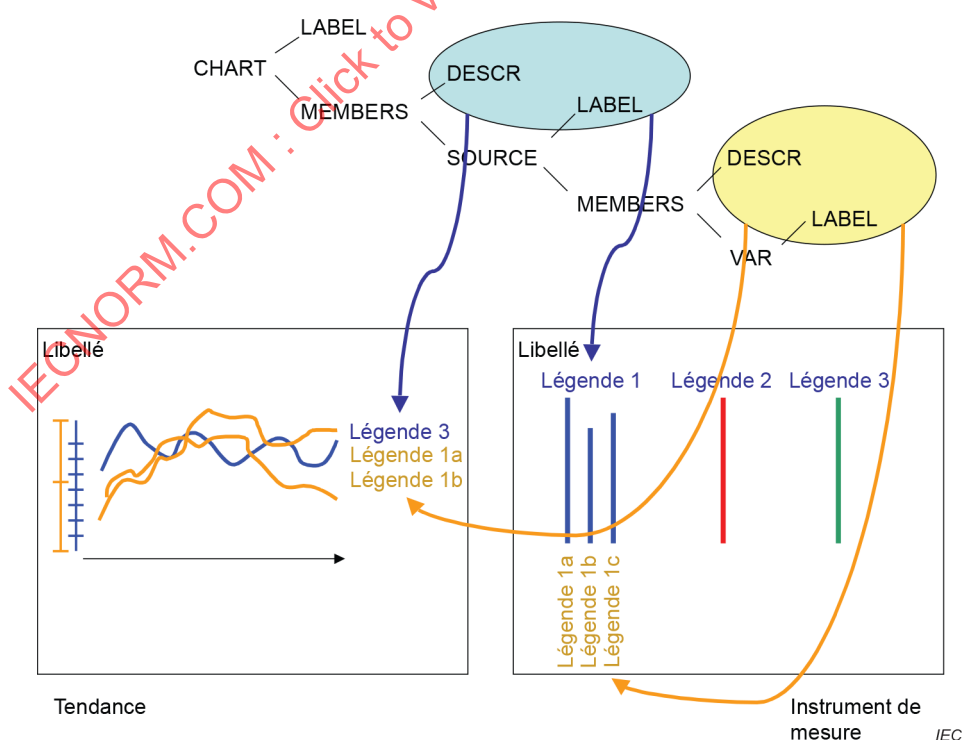


Figure 56 – Légende/aide pour les attributs SOURCE avec des attributs VARIABLE simples et multiples – Attribut Y_AXIS non défini

11.5.6.4 Tendances (attribut Y_AXIS défini) – Un attribut VARIABLE par attribut SOURCE et deux attributs SOURCE partagés (ou plus)

- a) Les règles de labellisation sont identiques à celles définies en 11.5.6.1.
- b) En outre, la légende de l'axe doit être affichée à l'aide du libellé AXIS.

11.5.6.5 Instruments de mesure (attribut Y_AXIS défini) – Un attribut VARIABLE par attribut SOURCE et deux attributs SOURCE partagés (ou plus)

- a) L'application EDD doit afficher les sources sur le même axe Y en les regroupant avec un instrument de mesure unique. Pour cet axe partagé, l'application EDD doit afficher chaque attribut SOURCE sous forme de barre verticale/horizontale distincte ou sous forme de jauge avec un instrument de mesure circulaire ou semi-circulaire unique.
- b) L'application EDD doit afficher la légende de chaque instrument de mesure (des différentes sources) d'un diagramme d'instruments de mesure (HORIZONTAL_BAR, VERTICAL_BAR et GAUGE) avec le libellé de l'attribut AXIS.
- c) En outre, l'application EDD doit afficher un libellé de légende/une aide à "barres/flèches" supplémentaire pour chaque barre avec la description/l'aide du membre de l'attribut CHART.
- d) Si la description/l'aide du membre de l'attribut CHART n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec le libellé/l'aide de la référence source.
- e) Si le libellé/l'aide de l'attribut SOURCE n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec la description/l'aide du membre de l'attribut SOURCE.
- f) Si la description/l'aide du membre de l'attribut SOURCE n'est pas définie ou est une chaîne vide (""), le libellé de la légende/l'aide doit s'afficher avec le libellé/l'aide de la référence de l'attribut VARIABLE (pour le libellé et l'aide d'une référence de variables, voir également Concaténation des libellés pour les références de variables indirectes et Concaténation des aides dans l' IEC 61804-4).

NOTE Les règles de labellisation pour 11.5.6.5 c), 11.5.6.5 d), 11.5.6.5 e) et 11.5.6.5 f) sont identiques à celles spécifiées en 11.5.6.1.

Se référer à la Figure 57, qui représente des exemples de légendes de diagrammes.

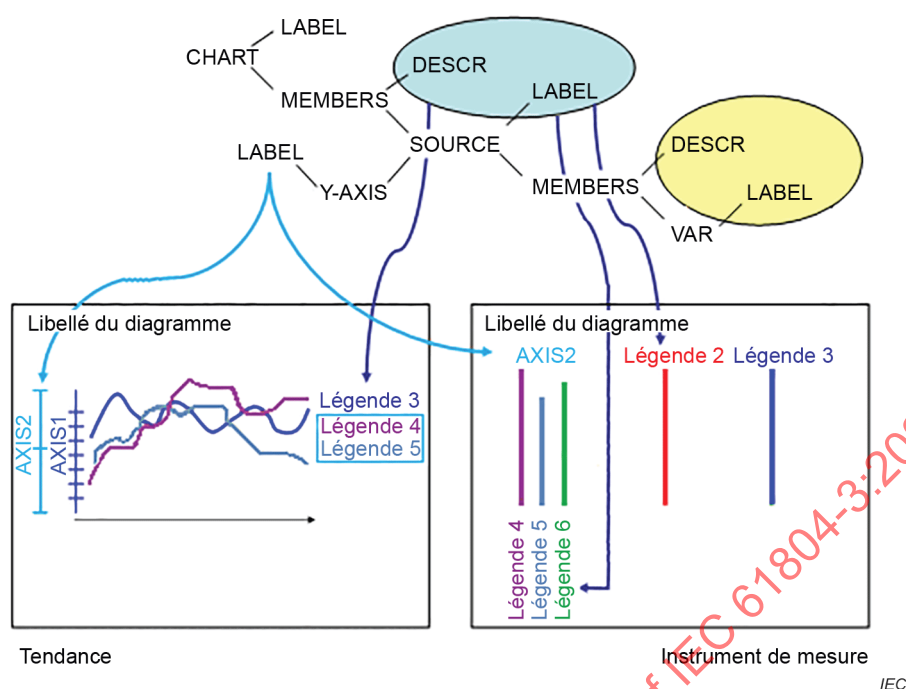


Figure 57 – Légende/aide pour les attributs SOURCE – Disposition de plusieurs attributs SOURCE avec un attribut Y_AXIS partagé

11.6 SOURCE

11.6.1 Structure générale

Objet

L'attribut SOURCE décrit une source de valeurs de données pour un attribut CHART. Les attributs SOURCE ont des attributs MEMBERS qui identifient la variable à présenter. L'attribut MEMBER doit avoir un nom et une référence à une valeur numérique. La description et l'aide sont des chaînes facultatives. Un attribut MEMBER peut être référencé depuis un autre emplacement dans l'EDD par l'identificateur de SOURCE et le nom de MEMBER spécifié en 16.13.

Il convient que les variables référencées dans l'attribut SOURCE soient définies dans les méthodes. La valeur peut être calculée à l'aide des variables de l'équipement et/ou des variables locales.

Si des attributs INIT_ACTIONS sont définis pour l'attribut SOURCE, l'application EDD invoque ces méthodes plutôt que de lire les variables.

Si des attributs REFRESH_ACTIONS sont définis pour l'attribut SOURCE, l'application EDD invoque ces méthodes de manière cyclique dans l'intervalle CYCLE_TIME plutôt que de lire les variables.

Pour les diagrammes de tendances (TYPE SCOPE, STRIP ou SWEEP), l'attribut VARIABLE de la liste MEMBER est représenté graphiquement dans le temps par rapport à l'attribut Y_AXIS indiqué. Pour les attributs CHART d'instruments de mesure (TYPE GAUGE, HORIZONTAL_BAR ou VERTICAL_BAR), la variable est présentée avec l'indicateur ou la barre (selon le cas).

Structure lexicale

```

SOURCE identifiant
{
    LABEL string-value;
    HELP string-value;
    EMPHASIS boolean-specifier;
    EXIT_ACTIONS { actions-specifier }
    INIT_ACTIONS { actions-specifier }
    LINE_COLOR line-color-specifier;
    LINE_TYPE line-type-specifier;
    MEMBERS { members-specifier }
    REFRESH_ACTIONS { actions-specifier }
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
    Y_AXIS [reference | reference<cond>];
}

```

Les attributs SOURCE sont spécifiés dans le Tableau 99.

Tableau 99 – Attributs SOURCE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	SOURCE		✓	✓	✓	✓	✓	✓	✓
14.5	EMPHASIS		o	o	o	o	o	o	o
11.6.2	EXIT_ACTIONS		o	o	o	o	o	o	o
14.8	HELP		o	o	o	o	o	o	o
11.6.3	INIT_ACTIONS		o	o	o	o	o	o	o
14.9	LABEL		o	o	o	o	o	o	o
14.10	LINE_COLOR		o	o	o	o	o	o	o
14.11	LINE_TYPE		o	o	o	o	o	o	o
14.12	MEMBERS		m	m	m	m	m	m	m
11.6.4	REFRESH_ACTIONS		o	o	o	o	o	o	o
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	x	o	o	o
11.6.5	Y_AXIS		o	o	o	o	o	o	o

Pour les attributs SOURCE ne spécifiant pas un attribut LINE_TYPE ou LINE_COLOR, chaque ligne doit adopter une apparence différente des autres dans l'attribut CHART.

11.6.2 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.

Structure lexicale

```
EXIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                [DEFINITION { compound-statement }]<cond>?
```

Les attributs EXIT_ACTIONS sont spécifiés dans le Tableau 108.

11.6.3 INIT_ACTIONS

Objet

L'attribut INIT_ACTIONS spécifie les actions qui sont exécutées avant l'affichage initial de la valeur de l'attribut SOURCE 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 l'attribut SOURCE et l'attribut EXIT_ACTIONS ne doit pas être exécuté si un attribut METHOD se termine pour une raison imprévue.

Indépendamment de l'affichage éventuel d'une source, l'attribut CHART doit s'afficher (l'attribut CHART peut être vide).

Structure lexicale

```
INIT_ACTIONS { actions-specifier }
actions-specifier:= [reference<cond> [, reference<cond>]*]?
                [DEFINITION { compound-statement }]<cond>?
```

Les attributs INIT_ACTIONS sont spécifiés dans le Tableau 108.

11.6.4 REFRESH_ACTIONS

Objet

L'attribut REFRESH_ACTIONS spécifie les actions qui sont exécutées avant la représentation initiale de l'attribut SOURCE, lorsque des modifications relatives aux attributs X_AXIS ou Y_AXIS sont apportées ou lorsqu'un attribut CYCLE_TIME est défini dans l'attribut CHART chaque fois que l'attribut 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, l'application EDD ne doit pas représenter graphiquement la valeur de l'attribut SOURCE si un attribut METHOD se termine pour une raison imprévue.

Indépendamment de l'affichage éventuel d'une source, l'attribut CHART doit s'afficher (l'attribut CHART peut être vide).

Il convient d'éviter les interfaces utilisateur intégrées à l'attribut METHODS REFRESH_ACTIONS. En effet, dans de nombreuses circonstances, l'actualisation peut survenir de manière fréquente et les interfaces utilisateur intégrées aux attributs REFRESH_ACTIONS peuvent créer des désagréments et des perturbations majeurs.

Structure lexicale

```
REFRESH_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*]|
                [DEFINITION { compound-statement }]<cond>
```

Les attributs REFRESH_ACTIONS sont spécifiés dans le Tableau 108.

11.6.5 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 fondé sur le temps, mais chaque attribut SOURCE sur un attribut CHART peut avoir son propre attribut Y_AXIS. Par défaut, l'affichage de l'axe Y d'un attribut CHART sans attribut Y_AXIS est déduit des données dans l'attribut SOURCE. Lorsque l'attribut Y_AXIS n'est pas spécifié, les valeurs minimales et maximales doivent être suffisamment importantes pour que toutes les données soient affichées. Les valeurs minimales et maximales doivent être modifiées en cas d'ajout de données à l'attribut CHART en dehors de la plage de l'attribut AXIS déduit.

Structure lexicale

Y_AXIS [*reference* | *reference*<cond>];

L'attribut Y_AXIS est spécifié dans le Tableau 100.

Tableau 100 – Attribut Y_AXIS

Attribut	Description
reference	Profils de communication: HART Référence à une instance d'AXIS.
reference<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Référence à une instance d'AXIS. Peut-être défini par une condition.

11.7 AXIS

11.7.1 Structure générale

Objet

L'attribut AXIS décrit l'axe d'un attribut CHART ou GRAPH, lequel spécifie la trame de référence qui guide la représentation graphique et l'interprétation des données. Par conséquent, les attributs MIN_VALUE, MAX_VALUE, SCALING et CONSTANT_UNIT associés à l'attribut AXIS sont des attributs importants.

L'axe est facultatif; lorsqu'il n'est pas déclaré, l'attribut GRAPH ou CHART détermine automatiquement les valeurs minimales et maximales de l'axe. Lorsqu'un axe n'est pas spécifié, les valeurs minimales et maximales de l'axe généré automatiquement doivent être suffisamment importantes pour que toutes les données soient affichées; les valeurs minimales et maximales doivent être modifiées en cas d'ajout de données à l'attribut CHART ou GRAPH en dehors de la plage de l'axe automatiquement produit. Le développeur de la DD peut inclure une référence à l'attribut AXIS s'il souhaite un contrôle approfondi.

Les attributs CHART et GRAPH sont cartésiens et disposent d'axes X et Y. Pour un attribut CHART ou GRAPH donné, l'axe x est commun à toutes les données représentées graphiquement; pour un attribut CHART, l'axe X doit être le temps. Un ou plusieurs axes Y peuvent être associés à un attribut CHART ou GRAPH. Lorsque toutes les données partagent un axe Y commun, la même construction AXIS doit être référencée dans toutes les déclarations de données (attribut WAVEFORM ou SOURCE). Plusieurs axes Y peuvent apparaître sur un même attribut CHART ou GRAPH. Dans ce cas, il convient que l'application EDD fournisse des indications permettant à l'opérateur d'associer l'attribut AXIS au tracé correspondant.

Structure lexicale

AXIS *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  CONSTANT_UNIT string-value;
  MAX_VALUE min-max-specifier;
  MIN_VALUE min-max-specifier;
  SCALING scaling-specifier;
}
```

Les attributs AXIS sont spécifiés dans le Tableau 101.

Tableau 101 – Attributs AXIS

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	AXIS		✓	✓	✓	✓	✓	✓	✓
11.7.4	CONSTANT_UNIT		○	○	○	○	○	○	○
14.8	HELP		○	○	○	○	○	○	○
14.9	LABEL		○	○	○	○	○	○	○
11.7.2	MAX_VALUE		○	○	○	○	○	○	○
11.7.2	MIN_VALUE		○	○	○	○	○	○	○
11.7.3	SCALING		○	○	○	○	○	○	○

11.7.2 MIN_VALUE, MAX_VALUE

Objet

L'attribut MIN_VALUE spécifie la valeur associée à la première marque de graduation de l'attribut AXIS. La première marque de graduation d'un attribut X_AXIS et d'un attribut Y_AXIS correspond respectivement à la marque de graduation la plus à gauche et à la marque de graduation la plus basse.

L'attribut MAX_VALUE spécifie la valeur associée à la dernière marque de graduation de l'attribut 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 à la marque de graduation la plus haute.

Par défaut, la valeur associée à la première ou la dernière marque de graduation de l'attribut AXIS est déduite des attributs WAVEFORM sur l'attribut GRAPH, si l'attribut MIN_VALUE ou MAX_VALUE n'est pas défini.

Les valeurs minimales/maximales de l'axe peuvent être référencées comme suit:

```
axis-identifiant.MIN_VALUE
axis-identifiant.MAX_VALUE
axis-identifiant.VIEW_MIN
axis-identifiant.VIEW_MAX
```

Les attributs VIEW_MIN et VIEW_MAX fournissent un accès aux valeurs minimales et maximales actuellement affichées sur l'attribut GRAPH ou CHART. Les attributs VIEW_MIN et VIEW_MAX sont disponibles, par exemple, à partir d'une méthode d'actualisation pour permettre à l'EDD de gérer les données de l'équipement en fonction du niveau de zoom actuel sur l'attribut GRAPH. Si le même attribut AXIS apparaît simultanément sur plusieurs attributs CHART ou GRAPH, le résultat de la référence VIEW_MIN/VIEW_MAX n'est pas défini et ne doit pas s'entendre comme fiable.

Structure lexicale

MAX_VALUE *min-max-specifier*;
MIN_VALUE *min-max-specifier*;
min-max-specifier := [*reference* | *double-constant* | *float-constant* | *integer-constant* | *unsigned-integer-constant*] <cond>

Les attributs MIN_VALUE et MAX_VALUE sont spécifiés dans le Tableau 102.

Tableau 102 – Attributs MIN_VALUE et MAX_VALUE

Type de valeurs	Description
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 numérique ou date/heure. 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.
double-constant	Constante à virgule flottante à double précision.
float-constant	Constante à virgule flottante à simple précision.
integer-constant	Constante entière.
unsigned-integer-constant	Constante entière non signée.

11.7.3 SCALING

Objet

L'attribut SCALING spécifie la manière dont il convient de mettre à l'échelle les marques de graduation de l'attribut AXIS. Par défaut, un attribut AXIS sans attribut SCALING est mis à l'échelle de façon linéaire. L'attribut SCALING doit être fixe lorsque l'attribut GRAPH ou CHART est affiché. Si l'attribut GRAPH ou CHART est fermé, puis rouvert, l'attribut AXIS peut obtenir un nouvel attribut SCALING.

Structure lexicale

SCALING *scaling-specifier*;
scaling-specifier := [**LINEAR** | **LOGARITHMIC**] <cond>

Les attributs SCALING sont spécifiés dans le Tableau 103.

Tableau 103 – Attribut SCALING

Valeur d'attribut	Description
LINEAR	Il convient que les marques de graduation sur l'axe soient mises à l'échelle de façon linéaire.
LOGARITHMIC	Il convient que les marques de graduation sur l'axe soient mises à l'échelle de façon logarithmique (base 10).

11.7.4 CONSTANT_UNIT

L'attribut CONSTANT_UNIT est utilisé si l'attribut AXIS a un code d'unité qui ne varie jamais. L'attribut CONSTANT_UNIT est spécifié comme une chaîne affichée avec la valeur. Un attribut AXIS sans unité constante ne comporte pas d'unité associée ou bien les unités ne sont pas constantes. Lorsque les unités techniques peuvent varier, elles doivent être représentées à l'aide d'une relation UNIT.

11.8 GRAPH

11.8.1 Structure générale

Objet

L'attribut GRAPH décrit un graphique utilisé pour afficher des données provenant d'un équipement dans l'application EDD. Un attribut GRAPH est utilisé pour afficher un ensemble de données statiques, par opposition à un attribut CHART qui est utilisé pour afficher des valeurs de données collectées en continu depuis un équipement. La série de données peut être stockée comme une matrice ou une liste dans l'équipement, ou il peut s'agir de données stockées pour l'équipement par l'application EDD avec la construction FILE. Dans chaque cas, un ou plusieurs attributs WAVEFORM peuvent être spécifiés par un graphique à des fins d'affichage par l'application EDD.

Chaque attribut MEMBER doit avoir un nom et une référence à un attribut WAVEFORM. La description et l'aide sont des chaînes facultatives; le cas échéant, elles sont utilisées pour fournir une légende à l'attribut GRAPH et une aide contextuelle concernant l'attribut WAVEFORM. Dans le cas contraire, il convient d'utiliser les attributs LABEL et HELP pour l'attribut WAVEFORM à cette fin. Lorsque l'application EDD commence à afficher un attribut GRAPH, les attributs INIT_ACTIONS et REFRESH_ACTIONS spécifiés pour l'attribut WAVEFORM sont exécutés. Chaque série de données est ensuite représentée graphiquement. L'application EDD permet à l'utilisateur de zoomer sur l'attribut GRAPH afin d'explorer les régions d'intérêt spécifiques. Le cas échéant, les nouveaux attributs VIEW_MIN et VIEW_MAX sont établis pour l'attribut AXIS et l'attribut REFRESH_ACTIONS est invoqué, ce qui permet à l'EDD de manipuler ou d'actualiser les données en fonction des exigences. L'application EDD doit également permettre à l'utilisateur d'obtenir une vue panoramique de la région d'affichage du graphique. La vue panoramique et le zoom ne sont restreints par aucune limite. Il convient qu'une légende étayant les formes d'ondes soit disponible pour guider l'utilisateur, avec les libellés et les codes d'unité relatifs à l'axe. Pour établir les légendes et les informations d'aide pour les courbes, l'application EDD doit prendre en charge les règles suivantes:

- l'application EDD doit afficher la description si elle est définie dans les membres de l'attribut GRAPH comme une légende. L'application EDD doit utiliser l'attribut LABEL de l'attribut WAVEFORM référencé si la description n'est pas définie. Une chaîne vide constitue une chaîne définie;
- l'application EDD doit fournir à l'utilisateur des informations d'aide pour les membres de l'attribut GRAPH. S'il n'existe aucun attribut HELP sur les membres de l'attribut GRAPH, les informations d'aide des attributs WAVEFORM doivent être utilisées;
- si l'attribut LABEL/HELP de l'attribut WAVEFORM n'est pas défini (ou est une chaîne vide), le libellé de la légende/l'aide doit s'afficher avec la description/l'aide du membre de l'attribut WAVEFORM;
- si la description/l'aide du membre de l'attribut WAVEFORM n'est pas définie (ou est une chaîne vide), le libellé de la légende/l'aide doit s'afficher avec une chaîne vide (vierge);
- toutes les légendes doivent être lisibles.

Une application EDD doit combiner des attributs WAVEFORM faisant référence au même attribut Y_AXIS. Les courbes doivent être représentées ensemble sur la région du graphique et un seul axe Y doit être représenté de chaque côté de la région du graphique.

Si les données de l'attribut WAVEFORM sont de CLASS CLASS DYNAMIC, il convient d'actualiser régulièrement l'attribut GRAPH.

L'attribut HANDLING d'un attribut WAVEFORM détermine si l'utilisateur peut modifier la forme d'onde. Lorsque l'attribut GRAPH est fermé, l'attribut EXIT_ACTIONS doit être invoqué pour permettre à l'EDD d'autoriser l'accès aux attributs WAVEFORM modifiés et leur manipulation. Il convient que l'application EDD fournisse un mécanisme permettant à l'utilisateur de modifier la forme d'onde (glisser-déposer direct, entrée de tableau, etc.) et un mécanisme permettant à l'utilisateur d'indiquer que la modification de la forme d'onde est achevée.

Pour un attribut GRAPH avec une définition CYCLE_TIME, l'application EDD doit suspendre la valeur source lue depuis l'équipement lorsque l'attribut GRAPH est modifié.

Il convient d'invoquer l'attribut POST_EDIT_ACTIONS de l'attribut VARIABLES pour chaque modification d'une forme d'onde de graphique. Les entrées de données peuvent être vérifiées et chaque donnée peut être modifiée selon la modification.

Les membres peuvent être référencés depuis un autre emplacement dans l'EDD par l'identificateur de GRAPH et le nom de MEMBER en utilisant la notation par point:

`graph-identifieur.member-name`

La couleur d'arrière-plan d'un affichage GRAPH est déterminée par l'application EDD. Afin de garantir une expérience commune, la couleur d'arrière-plan par défaut doit être WHITE. Il convient qu'une application EDD prenne en charge l'affichage simultané d'au moins six courbes.

La Figure 58 représente un graphique et les éléments visuels fournis par les constructions définies dans l'EDDL. Chaque élément est représenté plus précisément dans les paragraphes correspondants.

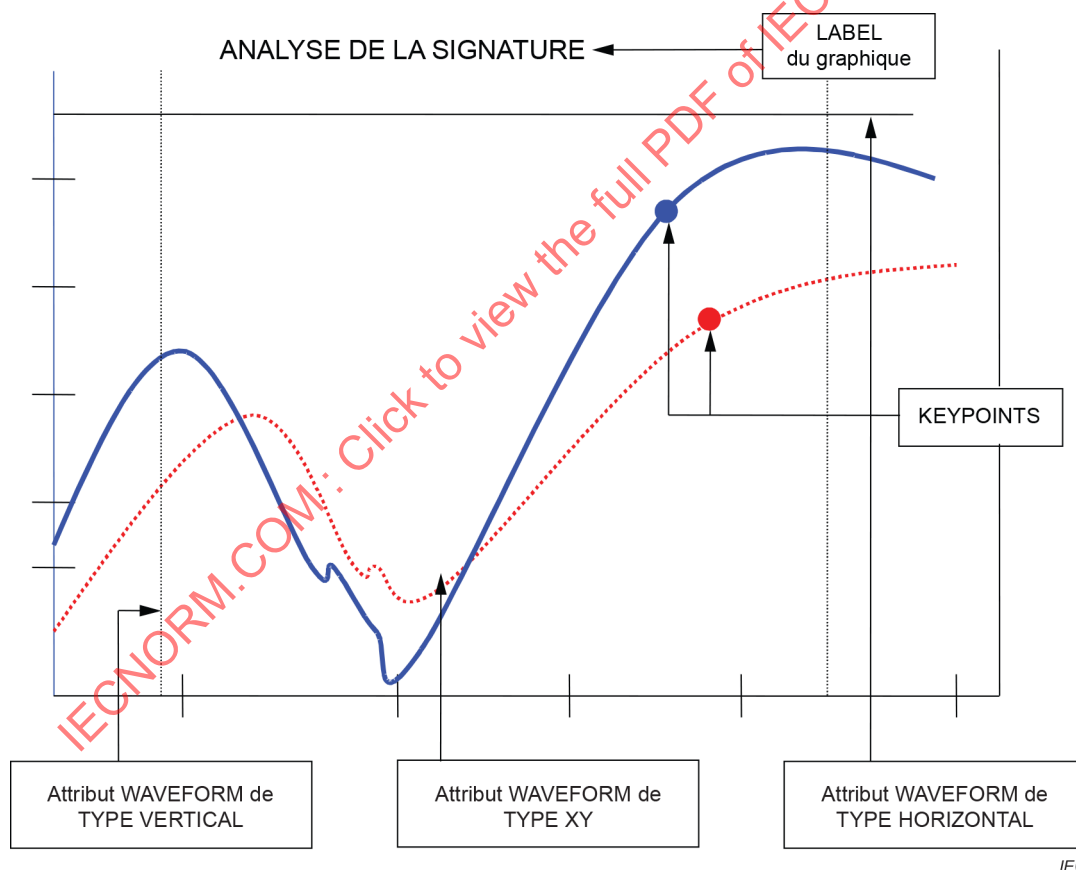


Figure 58 – Graphique et éléments visuels

Si une application EDDL ne prend pas en charge la représentation graphique d'un graphique, les valeurs des variables correspondantes peuvent être représentées dans un tableau.

Structure lexicale

GRAPH *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  CYCLE_TIME integer-expression | integer-expression<cond>;
  HANDLING handling-specifier;
  HEIGHT height-width-specifier;
  MEMBERS { members-specifier }
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
  WIDTH height-width-specifier;
  X_AXIS reference | reference<cond>;
}
```

Les attributs GRAPH sont spécifiés dans le Tableau 104.

Tableau 104 – Attributs GRAPH

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	GRAPH		✓	✓	✓	✓	✓	✓	✓
11.8.2		CYCLE_TIME	o	o	o	o	o	o	o
14.7		HEIGHT	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.12		MEMBERS	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	o	o	o	o
14.19		WIDTH	o	o	o	o	o	o	o
11.8.3		X_AXIS	o	o	o	o	o	o	o

Utilisation

La Figure 59 représente un exemple de graphique.

```

VARIABLE          x_value
{
    LABEL "...";
    HELP "...";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT [degC];
}

ARRAY             x_data
{
    LABEL "x-data";
    NUMBER_OF_ELEMENTS 1000;
    TYPE x_value;
}

VARIABLE          y_value
{
    LABEL "...";
    HELP "...";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT [degC];
}

ARRAY             y_data
{
    LABEL "y-data";
    NUMBER_OF_ELEMENTS 1000;
    TYPE y_value;
}

VARIABLE          x_min_value
{
    LABEL "...";
    HELP "...";
    CLASS LOCAL;
    TYPE FLOAT;
}

AXIS x_axis_signature
{
    LABEL "x axis";
    MIN_VALUE DEFAULT_X_MIN_VALUE;
    MAX_VALUE DEFAULT_X_MAX_VALUE;
}

AXIS y_axis_signature
{
    LABEL "y axis";
    MIN_VALUE DEFAULT_Y_MIN_VALUE;
    MAX_VALUE DEFAULT_Y_MAX_VALUE;
}

WAVEFORM value_signature1
{
    TYPE          XY
    {
        X_VALUES      { x_data }
        Y_VALUES      { y_data }
    }
    INIT_ACTIONS { read_first_signature }
    REFRESH_ACTIONS { read_signature }
    Y_AXIS y_axis_signature;
}

x_max_value      LIKE x_min_value;
device_x_min_value LIKE x_min_value;
device_x_max_value LIKE x_min_value;

y_min_value      LIKE x_min_value;
y_max_value      LIKE x_min_value;
device_y_min_value LIKE x_min_value;
device_y_max_value LIKE x_min_value;

METHOD read_first_signature
{
    DEFINITION

```

```

    {
        // lire les données de l'équipement en fonction de la valeur prédéfinie par
défaut
        // des attributs MIN_VALUE et MAX_VALUE des axes x et y
        x_min_value = DEFAULT_X_MIN_VALUE;
        x_max_value = DEFAULT_X_MAX_VALUE;
        y_min_value = DEFAULT_Y_MIN_VALUE;
        y_max_value = DEFAULT_Y_MAX_VALUE;
        WriteCommand (write_data_area);
        ReadCommand (read_data);
        x_axis_signature.MIN_VALUE = device_x_min_value;
        x_axis_signature.MAX_VALUE = device_x_max_value;
        y_axis_signature.MIN_VALUE = device_y_min_value;
        y_axis_signature.MAX_VALUE = device_y_max_value;
    }
}

METHOD read_signature
{
    DEFINITION
    {
        // lire les données de l'équipement en fonction des attributs MIN_VALUE et
MAX_VALUE actuels
        // de l'axe x
        x_min_value = x_axis_signature.VIEW_MIN;
        x_max_value = x_axis_signature.VIEW_MAX;
        y_min_value = y_axis_signature.VIEW_MIN;
        y_max_value = y_axis_signature.VIEW_MAX;
        WriteCommand (write_data_area);
        ReadCommand (read_data);
        if (device_x_min_value < x_min_value)
            x_axis_signature.VIEW_MIN = device_x_min_value;
        if (device_x_max_value > x_max_value)
            x_axis_signature.VIEW_MAX = device_x_max_value;
        if (device_y_min_value < y_min_value)
            y_axis_signature.VIEW_MIN = device_y_min_value;
        if (device_y_max_value > y_max_value)
            y_axis_signature.VIEW_MAX = device_y_max_value;
    }
}

GRAPH valve_signature
{
    MEMBERS
    {
        VAL_SIG, value_signature1;
    }
    X_AXIS x_axis_signature;
}

```

Figure 59 – Exemple de graphique

La Figure 60 représente un exemple d'EDD d'un axe commun pour les attributs WAVEFORM w1 et w2 et d'un second axe pour l'attribut WAVEFORM w3.

```

AXIS    x1 { }
AXIS    y1 { }
AXIS    y2 { }

GRAPH graph1
{
    MEMBERS
    {
        w1, w1;
        w2, w2;
        w3, w3;
    }
    X_AXIS x1;
}

WAVEFORM w1
{
    Y_AXIS y1;
    TYPE XY
    {
        Y_VALES {w1_values}
        X_INCREMENT 1
        X_INITIAL 0
    }
}

WAVEFORM w2
{
    Y_AXIS y1;
    TYPE XY
    {
        Y_VALES {w2_values}
        X_INCREMENT 1
        X_INITIAL 0
    }
}

WAVEFORM w3
{
    Y_AXIS y2;
    TYPE XY
    {
        Y_VALES {w3_values}
        X_INCREMENT 1
        X_INITIAL 0
    }
}

```

Figure 60 – Axes multiusages

Il convient que l'application EDD prenne en charge le zoom et le défilement sans prise en charge dans l'EDD. L'application EDD représente plus ou moins de points, ou les données de la partie actuellement non affichée de l'attribut WAVEFORM sur la zone d'affichage. En outre, l'attribut REFRESH_ACTIONS dans l'EDD peut prendre en charge le défilement et le zoom. L'application EDD invoque l'attribut REFRESH_ACTIONS si l'utilisateur fait défiler une zone qui n'est pas stockée dans les données de forme d'onde. Elle invoque également l'attribut REFRESH_ACTIONS si l'utilisateur effectue un zoom et qu'il convient de représenter plus de points. Les informations relatives à la position et au zoom peuvent être lues depuis l'axe avec les attributs VIEW_MIN et VIEW_MAX.

La Figure 61 représente un exemple d'EDD avec zoom et défilement pris en charge par l'équipement.

```

VARIABLE      y_value
{
    LABEL "...";
    HELP "...";
    CLASS DYNAMIC;
    TYPE FLOAT;
    CONSTANT_UNIT [degC];
}

ARRAY          y_data
{
    NUMBER_OF_ELEMENTS 1000;
    TYPE y_value;
}

COMMAND read_data
{
    SLOT ...; INDEX...;
    OPERATION READ;
    TRANSACTION
    {
        REPLY
        {
            device_t_min_value, device_t_max_value,
            device_y_min_value, device_y_max_value,
            y_data
        }
    }
}

COMMAND read_current_data
{
    SLOT ...; INDEX...;
    OPERATION READ;
    TRANSACTION
    {
        REPLY
        {
            device_t_min_value, device_t_max_value,
            device_y_min_value, device_y_max_value,
            y_data
        }
    }
}

COMMAND write_data_area
{
    SLOT ...; INDEX...;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            t_min_value, t_max_value,
            y_min_value, y_max_value
        }
    }
}

VARIABLE y_increment
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE TIME;
}

VARIABLE current_date_time1
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE DATE_AND_TIME;
}

VARIABLE t_min_value
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE DATE_AND_TIME;
}

```

```

}

t_max_value LIKE t_min_value;
t_device_min_value LIKE t_min_value;
t_device_max_value LIKE t_min_value;

VARIABLE y_min_value
{
    LABEL "..."; HELP "...";
    CLASS LOCAL;
    TYPE FLOAT;
}

y_max_value LIKE min_value;
device_y_min_value LIKE min_value;
device_y_max_value LIKE min_value;

AXIS y_axis_signature
{
    LABEL          "...";
    HELP           "...";
    MIN_VALUE      y_min_value;
    MAX_VALUE      y_max_value;
    SCALING        LINEAR;
}

AXIS t_axis_signature
{
    LABEL          "...";
    HELP           "...";
    MIN_VALUE      t_min_value;
    MAX_VALUE      t_max_value;
    SCALING        LINEAR;
}

WAVEFORM value_signature1
{
    LABEL          "...";
    HELP           "...";
    HANDLING       READ; // alternative aux attributs READ et WRITE
    LINE_TYPE      DATA1;
    EMPHASIS       TRUE;

    TYPE           XY
    {
        X_INITIAL      current_date_time1; // heure de début en millisecondes
        X_INCREMENT     y_increment;        // temps entre deux points en ms
        Y_VALUES        { y_data }
    }

    Y_AXIS          y_axis_signature;

    INIT_ACTIONS { init_signature}
    REFRESH_ACTIONS { refresh_signature}
}

METHOD init_signature
{
    DEFINITION
    {
        // lire les données actuelles et la zone de stockage t_device_min_value,
        // t_device_max_value, y_device_min_value, y_device_max_value
        read_command (read_current_data);

        // calcul des données de forme d'ondes
        value_signature1.X_INITIAL = t_device_max_value;
        value_signature1.X_INCREMENT = (t_device_max_value-t_device_min_value)/
            ( y_data.NUMBER_OF_ELEMENTS - 1 );

        // définir la zone visible sur la zone de l'équipement
        t_axis_signature.MIN_VALUE = t_device_min_value;
        t_axis_signature.MAX_VALUE = t_device_max_value;
        y_axis_signature.MIN_VALUE = y_device_min_value;
        y_axis_signature.MAX_VALUE = y_device_max_value;
    }
}

METHOD refresh_signature

```

```

{
    DEFINITION
    {
        // lire les données de l'équipement en fonction de la valeur actuelle
        // des attributs MIN_VALUE et MAX_VALUE de l'axe temps
        t_min_value = t_axis_signature.MIN_VALUE;

        t_max_value = t_axis_signature.MAX_VALUE;
        y_min_value = y_axis_signature.MIN_VALUE;
        y_max_value = y_axis_signature.MAX_VALUE;

        // écriture demandée pour t_min_value, t_max_value, y_min_value, y_max_value
        write_command (write_data_area);

        // point et zone de lecture      t_device_min_value, t_device_max_value,
        //                                y_device_min_value, y_device_max_value depuis
l'équipement
        read_command (read_data);

        // réduire la zone visible de la zone prise en charge par l'équipement
        if (t_device_min_value > t_min_value)
            t_axis_signature.MIN_VALUE = t_device_min_value;
        if (t_device_max_value < t_max_value)
            t_axis_signature.MAX_VALUE = t_device_max_value;
        if (y_device_min_value > y_min_value)
            y_axis_signature.MIN_VALUE = y_device_min_value;
        if (y_device_max_value < y_max_value)
            y_axis_signature.MAX_VALUE = y_device_max_value;
    }
}

```

Figure 61 – EDD avec zoom et défilement pris en charge par l'équipement

11.8.2 CYCLE_TIME

Objet

L'attribut CYCLE_TIME spécifie la durée, en millisecondes (ms), pendant laquelle l'application EDD doit attendre entre des lectures successives des valeurs sources de l'équipement. L'attribut GRAPH doit être actualisé (et l'attribut REFRESH_ACTIONS invoqué) selon la périodicité spécifiée par l'attribut CYCLE_TIME.

NOTE Pour des raisons de performance de l'application EDD (performances moindres d'une E/S multiplexée, par exemple), la fréquence de mise à jour, les latences et les giges peuvent entraîner une variation des durées de cycle réelles.

Par défaut, un attribut GRAPH n'a pas de durée de cycle. Dans ce cas, l'attribut CYCLE_TIME est par défaut défini sur

Profils de communication: CS, GPE, HART, PB, PN

+infinity (c'est-à-dire que l'attribut GRAPH n'est pas régulièrement mis à jour)

Profils de communication: FF, ISA100

1 000 millisecondes

Sinon, l'attribut CYCLE_TIME doit être résolu par une valeur positive. Même si l'attribut CYCLE_TIME peut faire référence à un attribut VARIABLE, l'application EDD doit conserver un attribut CYCLE_TIME fixe et invariable aussi longtemps que l'attribut GRAPH est affiché. L'attribut CYCLE_TIME doit être résolu et capturé immédiatement après l'exécution de l'attribut INIT_ACTIONS et avant l'affichage de l'attribut GRAPH.

L'attribut CYCLE_TIME doit être fixe lorsque l'attribut CHART est affiché. Si l'attribut CHART est fermé, puis rouvert, il peut obtenir un nouvel attribut CYCLE_TIME.

Structure lexicale

CYCLE_TIME *integer-expression* | *integer-expression*<cond>;

L'attribut CYCLE_TIME est spécifié dans le Tableau 105.

Tableau 105 – Attribut CYCLE_TIME

Type de valeurs	Description
integer-expression	Profils de communication: HART Temps, en millisecondes (ms), entre des lectures successives des valeurs sources de l'équipement.
integer-expression<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Temps, en millisecondes (ms), entre des lectures successives des valeurs sources de l'équipement. Peut-être défini par une condition.

11.8.3 X_AXIS

Objet

L'attribut X_AXIS spécifie la façon dont X_AXIS s'affiche. Chaque attribut WAVEFORM d'un attribut GRAPH partage un attribut X_AXIS commun, mais peut disposer de son propre attribut Y_AXIS ou partager le même attribut Y_AXIS. Par défaut, l'affichage de l'axe X d'un attribut GRAPH sans attribut X_AXIS est déduit des attributs WAVEFORM sur l'attribut GRAPH.

Structure lexicale

X_AXIS *reference* | *reference*<cond>;

L'attribut X_AXIS est spécifié dans le Tableau 106.

Tableau 106 – Attribut X_AXIS

Type de valeurs	Description
reference	Profils de communication: HART Référence à une instance d'AXIS.
reference<cond>	Référence à une instance de l'attribut AXIS; peut-être définie par une condition

11.8.4 Schéma de séquence pour les actions

La Figure 62 représente le schéma de séquence pour des actions relatives aux attributs GRAPH et aux membres de l'attribut WAVEFORM qu'ils contiennent. Les actions sont exécutées et achevées en séquence du haut vers le bas du schéma.

Les données provenant de l'équipement peuvent être ajustées avant d'être affichées par l'utilisation d'attributs INIT_ACTIONS et REFRESH_ACTIONS sur les membres de l'attribut WAVEFORM de l'attribut GRAPH. Tous les attributs INIT_ACTIONS devant être exécutés avant l'affichage de l'attribut GRAPH, l'attribut CYCLE_TIME peut être ajusté dans l'attribut INIT_ACTIONS sur les membres de l'attribut WAVEFORM.

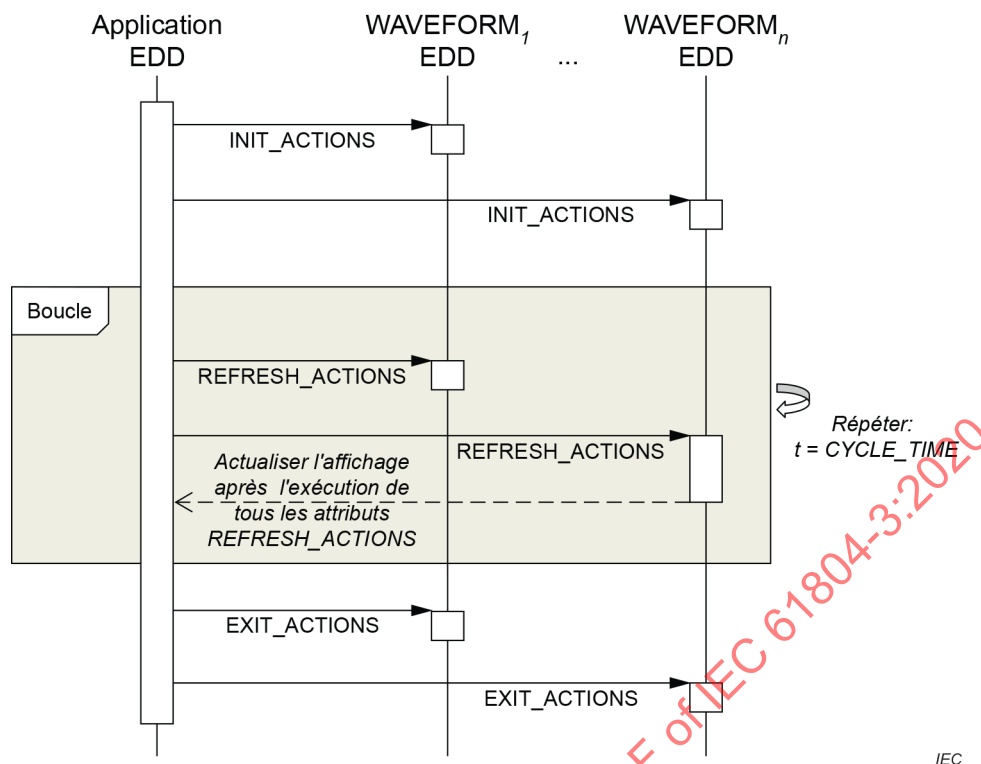


Figure 62 – Activation de l'attribut GRAPH

11.9 WAVEFORM

11.9.1 Structure générale

Objet

Un attribut WAVEFORM décrit un ensemble de données qui peut être affiché par un attribut GRAPH. L'attribut WAVEFORM est un vecteur ou une série de données décrit sous la forme de paires de données X-Y, une série de valeurs Y également espacées dans le sens horizontal (c'est-à-dire une série de données Y-T) ou une ligne horizontale ou verticale. Outre la fourniture même des données de forme d'onde, les points clés des séries de données peuvent être mis en évidence.

Structure lexicale

```

WAVEFORM identifier
{
    LABEL string-value;
    HELP string-value;
    EMPHASIS boolean-specifier;
    EXIT_ACTIONS { actions-specifier }
    HANDLING handling-specifier;
    INIT_ACTIONS { actions-specifier }
    KEY_POINTS
    {
        X_VALUES { key-points-specifier }
        Y_VALUES { key-points-specifier }
    }
    LINE_COLOR line-color-specifier;
    LINE_TYPE line-type-specifier;
    REFRESH_ACTIONS { actions-specifier }
    TYPE waveform-type-specifier;
    VALIDITY boolean-specifier;
    VISIBILITY boolean-specifier;
    Y_AXIS [reference | reference<cond>];
}
    
```

Les attributs WAVEFORM sont spécifiés dans le Tableau 107

Tableau 107 – Attributs WAVEFORM

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		WAVEFORM	✓	✓	✓	✓	✓	✓	✓
14.5		EMPHASIS	o	o	o	o	o	o	o
11.9.2		EXIT_ACTIONS	o	o	o	o	o	o	o
14.6		HANDLING	o	o	o	o	o	o	o
14.8		HELP	o	o	o	o	o	o	o
11.9.3		INIT_ACTIONS	o	o	o	o	o	o	o
11.9.4		KEY_POINTS	o	o	o	o	o	o	o
14.9		LABEL	o	o	o	o	o	o	o
14.10		LINE_COLOR	o	o	o	o	o	o	o
14.11		LINE_TYPE	o	o	o	o	o	o	o
11.9.5		REFRESH_ACTIONS	o	o	o	o	o	o	o
11.9.6		TYPE	m	m	m	m	m	m	m
14.17		VALIDITY	o	o	o	o	o	o	o
14.18		VISIBILITY	o	o	o	x	o	o	o
11.9.7		Y_AXIS	o	o	o	o	o	o	o

Pour les attributs WAVEFORM ne spécifiant pas un attribut LINE_TYPE ou LINE_COLOR, chaque ligne doit adopter une apparence différente des autres dans l'attribut GRAPH.

11.9.2 EXIT_ACTIONS

Objet

L'attribut EXIT_ACTIONS spécifie les actions qui sont exécutées lorsque l'attribut GRAPH contenant l'attribut WAVEFORM est fermé. Les actions de sortie permettent la capture et la manipulation des données de forme d'onde modifiées en fonction des besoins de l'EDD. En outre, si l'équipement a besoin se trouver dans un mode spécial pour fournir des données de forme d'onde, il est possible d'utiliser l'attribut EXIT_ACTION pour réinitialiser ce mode à la fermeture du graphique.

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.

Structure lexicale

```
EXIT_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs EXIT_ACTIONS sont spécifiés dans le Tableau 108.

11.9.3 INIT_ACTIONS

Objet

L'attribut INIT_ACTIONS spécifie les actions qui sont exécutées avant l'affichage initial de l'attribut WAVEFORM par un attribut GRAPH. Cette construction peut être utilisée non seulement pour la lecture initiale de l'attribut WAVEFORM depuis l'équipement ou depuis un attribut FILE, mais aussi pour les besoins de prétraitement (moyennage ou filtrage de données, par exemple), celui-ci étant effectué avant l'affichage.

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 l'attribut WAVEFORM si un attribut METHOD se termine pour une raison imprévue.

Indépendamment de l'affichage éventuel d'une forme d'onde, l'attribut CHART doit s'afficher (l'attribut GRAPH peut être vide).

Structure lexicale

```
INIT_ACTIONS { actions-spezifier }
actions-spezifier:= [reference<cond> [, reference<cond>]*]?
                    [DEFINITION { compound-statement }]<cond>?
```

Les attributs INIT_ACTIONS sont spécifiés dans le Tableau 108.

Tableau 108 – Attributs EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS

Type de valeurs	Description
reference	Référence à une instance de METHOD ou à une instance de METHOD avec arguments.
DEFINITION	Profils de communication: CS, GPE, PB, PN Élément lexical (voir 14.4).

11.9.4 KEY_POINTS

11.9.4.1 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. Il n'est pas nécessaire que les points clés correspondent directement aux points de données spécifiés par l'attribut TYPE. La façon dont ces points sont mis en évidence est définie par l'application EDD. Par défaut, il convient de ne mettre aucun point d'un attribut WAVEFORM sans attribut KEY_POINTS en évidence.

Structure lexicale

KEY_POINTS

```
{
  X_VALUES { values-spezifier }
  Y_VALUES { values-spezifier }
}
```

Les attributs KEY_POINTS sont spécifiés dans le Tableau 109.

Tableau 109 – Attribut KEY_POINTS

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
11.9.4.2	X_VALUES	m	m	m	m	m	m	m
11.9.4.2	Y_VALUES	m	m	m	m	m	m	m

11.9.4.2 X_VALUES / Y_VALUES

Objet

Les attributs X_VALUES et Y_VALUES spécifient les coordonnées X et Y de chaque point devant être mis en évidence. Chaque coordonnée X spécifiée dans X_VALUES doit correspondre à une coordonnée Y spécifiée dans Y_VALUES.

Structure lexicale

```
X_VALUES { values-spezifier }
Y_VALUES { values-spezifier }
values-spezifier:= [reference | double-constant | float-constant |
                    integer-constant | unsigned-integer-constant]+
```

Les attributs X_VALUES et Y_VALUES sont spécifiés dans le Tableau 110.

Tableau 110 – Attributs X_VALUES et Y_VALUES

Type de valeurs	Description
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.
double	Constante à virgule flottante à double précision.
float	Constante à virgule flottante à simple précision.
integer	Constante entière.
unsigned_integer	Constante entière non signée.

11.9.5 REFRESH_ACTIONS

Objet

L'attribut REFRESH_ACTIONS spécifie les actions qui sont exécutées avant la représentation initiale de l'attribut WAVEFORM, lorsque des modifications relatives aux attributs X_AXIS ou Y_AXIS sont apportées ou lorsqu'un attribut CYCLE_TIME est défini dans l'attribut CHART chaque fois que l'attribut CYCLE_TIME s'écoule.

Dans la méthode, la zone visible peut être calculée et définie sur l'axe avec les attributs VIEW_MIN et VIEW_MAX. Tous les attributs WAVEFORM associés au même attribut AXIS doivent être représentés dans la même zone de l'attribut GRAPH.

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 l'attribut WAVEFORM doivent être restaurés à leurs valeurs d'origine par l'application EDD, et l'attribut WAVEFORM de l'attribut GRAPH ne doit pas être affiché.

Indépendamment de l'affichage éventuel d'une forme d'onde, l'attribut CHART doit s'afficher (l'attribut GRAPH peut être vide).

Il convient d'éviter les interfaces utilisateur intégrées à l'attribut METHODS REFRESH_ACTIONS. En effet, dans de nombreuses circonstances, l'actualisation peut survenir de manière fréquente et les interfaces utilisateur intégrées aux attributs REFRESH_ACTIONS peuvent créer des désagréments et des perturbations majeurs.

Structure lexicale

```
REFRESH_ACTIONS { [actions-specifier [, actions-specifier]*] }
actions-specifier:= [reference<cond> [, reference<cond>]*] |
                    [DEFINITION { compound-statement }]<cond>
```

Les attributs REFRESH_ACTIONS sont spécifiés dans le Tableau 108.

11.9.6 TYPE

11.9.6.1 Structure générale

Objet

L'attribut TYPE spécifie le type des données contenues dans l'attribut WAVEFORM.

Structure lexicale

TYPE *waveform-type-specifier*;

waveform-type-specifier := [XY | YT | HORIZONTAL | VERTICAL]

Les attributs TYPE sont spécifiés dans le Tableau 111.

Tableau 111 – Attribut TYPE

Référence	Attribut	Profils de communication						
		CS	FF	GPE	HART	ISA100	PB	PN
11.9.6.4	HORIZONTAL	S	S	S	S	S	S	S
11.9.6.5	VERTICAL	S	S	S	S	S	S	S
11.9.6.2	XY	S	S	S	S	S	S	S
11.9.6.3	YT	S	S	S	S	S	S	S

11.9.6.2 XY

Objet

L'attribut XY spécifie un attribut WAVEFORM qui contient une liste de points (x,y). Le type WAVEFORM XY permet la représentation graphique d'un ensemble arbitraire de points de données. Il n'est pas exigé que les données soient uniformément espacées.

Structure lexicale

XY

```
{
  X_VALUES { values-specifier }
  Y_VALUES { values-specifier }
  NUMBER_OF_POINTS integer-expression<cond>;
}
```

Les attributs XY sont spécifiés dans le Tableau 112.

Tableau 112 – Attribut XY

Utilisation	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. L'attribut X_VALUES est spécifié en 11.9.4.2.
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. L'attribut Y_VALUES est spécifié en 11.9.4.2.
o	NUMBER_OF_POINTS	Spécifie le nombre de points de données valides dans les attributs X_VALUES et Y_VALUES. Une série de données représente fréquemment un ensemble échantillonné de données acquises par l'équipement. En fonction de la configuration et du fonctionnement actuel de l'équipement, le nombre de points acquis par l'équipement peut varier. Les applications EDD ne doivent pas représenter graphiquement plus de points que ceux indiqués par l'attribut NUMBER_OF_POINTS. Par défaut, le nombre de points dans l'attribut WAVEFORM sans attribut NUMBER_OF_POINTS est égal à la taille des attributs X_VALUES et Y_VALUES.

11.9.6.3 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. Le type WAVEFORM XY permet la représentation graphique d'un ensemble de données uniformément séparées ou échantillonnées.

Le type WAVEFORM XY doit inclure les attributs X_INITIAL, X_INCREMENT et Y_VALUES. La coordonnée Y de chaque point de données est spécifiée dans la liste Y_VALUES. La coordonnée X correspondante est calculée avec les attributs X_INITIAL et X_INCREMENT, et le décalage (index) dans la liste Y_VALUES. En d'autres termes, les coordonnées du i^{e} point de données peuvent être calculées:

```
y-coordinate = Y_VALUES[i];
x-coordinate = X_INITIAL + (X_INCREMENT * i);
```

Structure lexicale

YT

```
{
  X_INITIAL [reference | double-expression | float-expression |
             integer-expression | unsigned-integer-expression]<cond>;
  X_INCREMENT [reference | double-expression | float-expression |
              integer-expression | unsigned-integer-expression]<cond>;
  Y_VALUES { values-specifier }
  NUMBER_OF_POINTS integer-expression<cond>;
}
```

Les attributs YT sont spécifiés dans le Tableau 113.

Tableau 113 – Attributs YT

Utilisation	Attribut	Description
m	X_INITIAL	Spécifie la coordonnée X du premier point dans l'attribut WAVEFORM. 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. 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. L'attribut Y_VALUES est spécifié en 11.9.4.2.
o	NUMBER_OF_POINTS	Spécifie le nombre de points de données valides dans X_VALUES. Une série de données représente fréquemment un ensemble échantillonné de données acquises par l'équipement. En fonction de la configuration et du fonctionnement actuel de l'équipement, le nombre de points acquis par l'équipement peut varier. Les applications EDD ne doivent pas représenter graphiquement plus de points que ceux indiqués par l'attribut NUMBER_OF_POINTS. Par défaut, le nombre de points dans l'attribut WAVEFORM sans attribut NUMBER_OF_POINTS est égal à la taille de l'attribut X_VALUES.

11.9.6.4 HORIZONTAL

Objet

L'attribut HORIZONTAL spécifie un attribut WAVEFORM qui contient des lignes horizontales.

Structure lexicale

HORIZONTAL

```
{
  Y_VALUES { values-spezifier }
}
```

L'attribut HORIZONTAL est spécifié dans le Tableau 114.

Tableau 114 – Attribut HORIZONTAL

Utilisation	Attribut	Description
m	Y_VALUES	Spécifie la coordonnée Y de chaque ligne horizontale dans l'attribut WAVEFORM. L'attribut Y_VALUES est spécifié en 11.9.4.2.

11.9.6.5 VERTICAL

Objet

L'attribut VERTICAL spécifie un attribut WAVEFORM qui contient des lignes verticales.

Structure lexicale

VERTICAL

```
{
  X_VALUES { values-spezifier }
}
```

L'attribut VERTICAL est spécifié dans le Tableau 115.

Tableau 115 – Attribut VERTICAL

Utilisation	Attribut	Description
m	X_VALUES	Spécifie la coordonnée X de chaque ligne verticale dans l'attribut WAVEFORM. L'attribut X_VALUES est spécifié en 11.9.4.2.

11.9.7 Y_AXIS

Objet

L'attribut Y_AXIS spécifie le mode d'affichage de l'axe Y de l'attribut WAVEFORM. Chaque attribut WAVEFORM d'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 déduit des données dans l'attribut WAVEFORM. Lorsque l'attribut Y_AXIS n'est pas spécifié, les valeurs minimales et maximales doivent être suffisamment importantes pour que toutes les données soient affichées. Les valeurs minimales et maximales doivent être modifiées en cas d'ajout de données à l'attribut GRAPH en dehors de la plage de l'attribut AXIS déduit.

Structure lexicale

Y_AXIS [*reference* | *reference*<cond>];

L'attribut Y_AXIS est spécifié dans le Tableau 116.

Tableau 116 – Attribut Y_AXIS

Attribut	Description
reference	Profils de communication: HART Référence à une instance d'AXIS.
reference<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Référence à une instance d'AXIS. Peut-être défini par une condition.

11.10 PLUGIN

11.10.1 Structure générale

Objet

PLUGIN décrit un plug-in d'interface utilisateur (UIP). Cela permet de référencer un point d'entrée externe depuis le DD à l'aide d'un UUID.

Structure lexicale

PLUGIN *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  UUID uuid-specifier;
  VALIDITY boolean-specifier;
  VISIBILITY boolean-specifier;
}
```

Les attributs PLUGIN sont spécifiés dans le Tableau 117.

Tableau 117 – Attributs PLUGIN

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	PLUGIN		✓	✓	✓	✓	✓	✓	✓
14.8	HELP		o	o	o	o	o	o	o
14.9	LABEL		m	m	m	m	m	m	m
11.10.2	UUID		m	m	m	m	m	m	m
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	o	o	o	o

11.10.2 UUID

Objet

L'attribut UUID spécifie l'identificateur unique utilisé pour identifier l'attribut PLUGIN. L'attribut UUID est un identificateur universel unique spécifié dans l'ISO/IEC 9834-8. L'UUID est spécifié par une chaîne de 36 caractères constituée de 32 chiffres hexadécimaux (non sensibles à la casse) décomposés en cinq groupes séparés par des tirets (respectivement 8 chiffres hexadécimaux suivis de 4, 4, 4 et 12 chiffres hexadécimaux). Par exemple:

3F99B900-4D87-101B-99B7-AA0004007F07

NOTE Cet attribut peut ne pas être conditionnel.

Structure lexicale

UUID *uuid-spezifier*;

L'attribut UUID est spécifié dans le Tableau 118.

Tableau 118 – Attribut UUID

Type de valeurs	Description
uuid-spezifier	Identificateur unique du plug-in spécifié dans le manifeste.

12 METHOD

12.1 Structure générale

Objet

Un attribut METHOD est utilisé pour définir un sous-programme qui est exécuté dans l'application EDD.

NOTE 1 Un attribut METHOD est utilisé pour spécifier des contrôles de cohérence, des algorithmes de conversion entre des variables EDD, des accusés de réception utilisateur ou des interactions d'équipement spécifiques (procédures opérationnelles normalisées pour la maintenance, l'étalonnage, la mise en service ou la configuration, par exemple).

NOTE 2 Les méthodes ne sont pas destinées à être utilisées pour améliorer les fonctionnalités des équipements ou pour générer de nouvelles données d'équipement. En d'autres termes, des méthodes permettant de générer des processus ou des états, qu'il convient de fournir par le biais de l'équipement, empêchent l'utilisation homogène des données hors des applications EDD.

Structure lexicale

```
METHOD identifier [( parameter-declaration [, parameter-declararion]* )]?
{
    LABEL string-value;
    HELP string-value;
    ACCESS [OFFLINE | ONLINE];
    CLASS method-class-spezifier [& method-class-spezifier]*;
    DEFINITION { compound-statement }
    PRIVATE boolean-spezifier;
    TYPE method-type-spezifier;
    VALIDITY boolean-spezifier;
    VISIBILITY boolean-spezifier;
}
parameter-declaration := parameter-type [&]?identifier[[]]?
```

Les attributs METHOD sont spécifiés dans le Tableau 119.

L'attribut parameter-type est spécifié dans le Tableau 120.

Tableau 119 – Attributs METHOD

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	METHOD		✓	✓	✓	✓	✓	✓	✓
12.2	ACCESS		o	x	o	x	x	o	o
12.3	CLASS		o	o	o	o	o	o	o
14.4	DEFINITION		m	m	m	m	m	m	m
14.8	HELP		o	o	o	o	o	o	o
14.9	LABEL		o	o	o	o	o	o	o
14.13	PRIVATE		o	o	o	o	o	o	o
12.4	TYPE		o	o	o	o	o	o	o
14.17	VALIDITY		o	o	o	o	o	o	o
14.18	VISIBILITY		o	o	o	o	o	o	o

Tableau 120 – Types de paramètres

Valeur d'attribut	Description
char	L'attribut METHOD attend un paramètre de type char.
unsigned char	L'attribut METHOD attend un paramètre de type unsigned char.
short	L'attribut METHOD attend un paramètre de type short.
unsigned short	L'attribut METHOD attend un paramètre de type unsigned short.
int	L'attribut METHOD attend un paramètre de type int.
unsigned int	L'attribut METHOD attend un paramètre de type unsigned int.
long	L'attribut METHOD attend un paramètre de type long.
unsigned long	L'attribut METHOD attend un paramètre de type unsigned long.
long long	L'attribut METHOD attend un paramètre de type long long.
unsigned long long	L'attribut METHOD attend un paramètre de type unsigned long long.
float	L'attribut METHOD attend un paramètre de type float.
double	L'attribut METHOD attend un paramètre de type double.
time_t	L'attribut METHOD attend un paramètre de type time_t.
DD_STRING	L'attribut METHOD attend un paramètre de type DD_STRING.
DD_ITEM	L'attribut METHOD attend un paramètre de type DD_ITEM.

12.2 ACCESS

Objet

L'attribut ACCESS spécifie si les valeurs de VARIABLE stockées dans l'équipement 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 121.

Tableau 121 – Attributs ACCESS

Valeur d'attribut	Description
OFFLINE	Les valeurs des variables utilisées dans la définition de METHOD sont lues ou écrites depuis/vers l'ensemble de données hors connexion.
ONLINE	Les valeurs des variables utilisées dans la définition de METHOD sont effectivement lues depuis l'équipement. Les changements de valeurs de VARIABLE sont écrits avec des builtins.

12.3 CLASS

Objet

L'attribut CLASS spécifie l'effet d'un attribut METHOD sur un équipement. Cet attribut est destiné à être utilisé par l'application EDD pour implémenter des niveaux d'autorisation et définir comment les attributs METHOD sont présentés.

Structure lexicale

CLASS *method-class-spezifier* [*& method-class-spezifier*]*;
:= [ALARM | ANALOG_OUTPUT | BACKGROUND | COMPUTATION | CONTAINED |
CORRECTION | DEVICE | DIAGNOSTIC | DISCRETE | FREQUENCY | HART | INPUT |
LOCAL_DISPLAY | OPERATE | OUTPUT | SERVICE | SPECIALIST | TUNE]

Les attributs CLASS sont spécifiés dans le Tableau 122.

Tableau 122 – Attributs CLASS

Attribut	Description	Profils de communication
ALARM	L'attribut METHOD est associé à une alarme (par exemple, spécification de limites d'alarme, indication d'état d'alarme, etc.)	CS, FF, GPE, ISA100, PB, PN
ANALOG_OUTPUT	L'attribut METHOD fait partie d'un bloc de sortie analogique.	CS, GPE, HART, PB, PN
BACKGROUND	L'attribut CLASS BACKGROUND est fourni pour améliorer la simulation d'un équipement avec son EDD. Dans toutes les applications EDD différentes des simulateurs, cette classe doit être ignorée. L'attribut METHODS CLASS BACKGROUND ne doit appartenir à aucun autre attribut CLASS. L'attribut CLASS BACKGROUND doit uniquement être utilisé dans les attributs METHODS et, le cas échéant, indiquer que le simulateur doit régulièrement exécuter l'attribut METHOD. Tous les attributs METHODS CLASS BACKGROUND doivent être exécutés dans la période monotone spécifiée par l'attribut VARIABLE background_period. L'attribut METHOD ne doit pas être exécuté lors d'une opération de manipulation de paquets (c'est-à-dire entre la réception d'un paquet de demandes et la distribution du paquet de réponses correspondant au gestionnaire de communication). Si l'attribut background_period expire pendant cet intervalle, l'attribut METHOD doit être exécuté dès que la réponse de commande est générée.	HART
COMPUTATION	L'attribut METHOD fait partie d'un bloc de calcul.	CS, GPE, HART, PB, PN
CONTAINED	L'attribut METHOD représente les caractéristiques physiques de l'équipement.	CS, GPE, PB, PN
CORRECTION	L'attribut METHOD prend en charge les transducteurs dans l'équipement. L'attribut METHOD peut établir des limites de transducteur, fournir un amortissement de signal, indiquer la valeur du transducteur, linéariser ou étalonner le transducteur, etc.	CS, GPE, HART, PB, PN

Attribut	Description	Profils de communication
DEVICE	L'attribut METHOD spécifie les caractéristiques physiques de l'équipement.	CS, GPE, HART, PB, PN
DIAGNOSTIC	L'attribut METHOD évalue l'état de l'équipement.	CS, FF, GPE, HART, ISA100, PB, PN
DISCRETE	L'attribut METHOD prend en charge l'E/S discrète et/ou numérique de l'équipement.	CS, GPE, HART
FREQUENCY	L'attribut METHOD prend en charge l'E/S de fréquence de l'équipement.	CS, GPE, HART
HART	L'attribut METHOD fait partie du bloc HART qui caractérise l'interface HART. Il y a un seul bloc HART.	CS, HART
INPUT	L'attribut 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 d'un bloc d'entrée peuvent être déterminés par la sortie d'un autre bloc.	CS, FF, GPE, HART, ISA100, PB, PN
LOCAL_DISPLAY	L'attribut 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'équipement.	CS, GPE, HART, PB, PN
OPERATE	L'attribut METHOD est utilisé pour contrôler le fonctionnement d'un bloc.	CS, FF, GPE, PB, PN
OUTPUT	L'attribut METHOD fait partie du bloc de sortie. Les valeurs de paramètres de sortie peuvent être accessibles par une autre entrée de bloc.	CS, FF, GPE, ISA100, PB, PN
SERVICE	L'attribut METHOD est utilisé lors de la conduite de la maintenance de routine.	CS, FF, GPE, HART, ISA100, PB, PN
SPECIALIST	L'attribut METHOD doit être uniquement exécutable pour un utilisateur spécialiste de l'application EDD. Avec cet attribut CLASS, l'attribut METHOD ne doit pas être exécutable pour un utilisateur non spécialiste. Si VALIDITY ou VISIBILITY est FALSE, l'attribut METHOD ne doit pas être visible. Si l'attribut n'est pas défini, l'attribut METHOD doit être exécutable.	CS, GPE, HART, ISA100, PB, PN
TUNE	L'attribut METHOD est utilisé pour régler l'algorithme d'un bloc.	CS, FF, GPE, ISA100, PB, PN
Profils de communication: HART Seuls les attributs SPECIALIST et BACKGROUND sont recommandés à des fins d'utilisation. Les autres attributs CLASS pris en charge par le profil sont réservés à des fins de rétrocompatibilité et leur utilisation n'est pas recommandée pour les nouvelles conceptions.		
Profils de communication: FF, ISA100 Une et une seule des classes suivantes: INPUT, OUTPUT. Zéro ou plusieurs des classes restantes prises en charge par le profil.		

12.4 TYPE

Objet

L'attribut TYPE spécifie le type de la valeur retournée.

Profils de communication: HART

L'attribut TYPE d'un attribut METHOD doit uniquement être déclaré pour les attributs METHOD exclusivement référencés à partir d'autres attributs METHOD (c'est-à-dire qu'une référence d'un attribut MENU vers un attribut METHOD ayant un attribut TYPE déclaré n'est pas admise).

Structure lexicale

TYPE *method-type-specifier*;

method-type-specifier := [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 123.

Tableau 123 – Attributs TYPE

Attribut	Description
char	L'attribut METHOD retourne une valeur de type char.
unsigned char	L'attribut METHOD retourne une valeur de type unsigned char.
short	L'attribut METHOD retourne une valeur de type short.
unsigned short	L'attribut METHOD retourne une valeur de type unsigned short.
int	L'attribut METHOD retourne une valeur de type int.
unsigned int	L'attribut METHOD retourne une valeur de type unsigned int.
long	L'attribut METHOD retourne une valeur de type long.
unsigned long	L'attribut METHOD retourne une valeur de type unsigned long.
long long	L'attribut METHOD retourne une valeur de type long long.
unsigned long long	L'attribut METHOD retourne une valeur de type unsigned long long.
float	L'attribut METHOD retourne une valeur de type float.
double	L'attribut METHOD retourne une valeur de type double.
time_t	L'attribut METHOD retourne une valeur de type time_t.
DD_STRING	L'attribut METHOD retourne une valeur de type DD_STRING.

13 Modélisation d'équipement

13.1 BLOCK_A

13.1.1 Structure générale

Objet

Un équipement peut être organisé en blocs logiques. Chaque attribut BLOCK_A est un regroupement logique d'attributs PARAMETERS et d'informations supplémentaires pour applications BLOCK_A. Les types de blocs 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 *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  AXIS_ITEMS { reference [, reference]* }
  CHARACTERISTICS reference;
  CHARTS { [identifier , reference [, description [, help]? ]?;]+ }
  CHART_ITEMS { reference [, reference]* }
  COLLECTION_ITEMS { reference [, reference]* }
  EDIT_DISPLAY_ITEMS { reference [, reference]* }
  FILES { [identifier , reference [, description [, help]? ]?;]+ }
  FILE_ITEMS { reference [, reference]* }
  GRAPHS { [identifier , reference [, description [, help]? ]?;]+ }
  GRAPH_ITEMS { reference [, reference]* }
  GRIDS { [identifier , reference [, description [, help]? ]?;]+ }
  GRID_ITEMS { reference [, reference]* }
  IMAGE_ITEMS { reference [, reference]* }
  ITEM_ARRAY_ITEMS { reference [, reference]* }
  LISTS { [identifier , reference [, description [, help]? ]?;]+ }
  LIST_ITEMS { reference [, reference]* }
  LOCAL_PARAMETERS { [identifier , reference [, description [,
help]? ]?;]+ }
  MENUS { [identifier , reference [, description [, help]? ]?;]+ }
  MENU_ITEMS { reference [, reference]* }
  METHODS { [identifier , reference [, description [, help]? ]?;]+ }
  METHOD_ITEMS { reference [, reference]* }
  PARAMETERS { [identifier , reference [, description [, help]? ]?;]+ }
  PARAMETER_LISTS { [identifier , reference [, description [,
help]? ]?;]+ }
  PLUGINS { [identifier , reference [, description [, help]? ]?;]+ }
  PLUGIN_ITEMS { reference [, reference]* }
  REFRESH_ITEMS { reference [, reference]* }
  SEMANTIC_MAP_ITEMS { reference [, reference]* }
  SOURCE_ITEMS { reference [, reference]+ }
  UNIT_ITEMS { reference [, reference]* }
  WAVEFORM_ITEMS { reference [, reference]* }
  WRITE_AS_ONE_ITEMS { reference [, reference]* }
}
```

Les attributs BLOCK A sont spécifiés dans le Tableau 124.

Tableau 124 – Attributs BLOCK_A

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	BLOCK_A		x	✓	x	x	✓	x	x
13.1.2	AXIS_ITEMS		x	o	x	x	o	x	x
13.1.3	CHARACTERISTICS		x	m	x	x	x	x	x
13.1.4	CHARTS		x	o	x	x	o	x	x
13.1.5	CHART_ITEMS		x	o	x	x	o	x	x
13.1.6	COLLECTION_ITEMS		x	o	x	x	o	x	x
13.1.7	EDIT_DISPLAY_ITEMS		x	o	x	x	o	x	x
13.1.8	FILES		x	o	x	x	o	x	x
13.1.9	FILE_ITEMS		x	o	x	x	o	x	x
13.1.10	GRAPHS		x	o	x	x	o	x	x
13.1.11	GRAPH_ITEMS		x	o	x	x	o	x	x
13.1.12	GRIDS		x	o	x	x	o	x	x
13.1.13	GRID_ITEMS		x	o	x	x	o	x	x
14.8	HELP		x	o	x	x	o	x	x
13.1.14	IMAGE_ITEMS		x	o	x	x	o	x	x
13.1.15	ITEM_ARRAY_ITEMS		x	o	x	x	o	x	x
14.9	LABEL		x	m	x	x	m	x	x
13.1.16	LISTS		x	o	x	x	o	x	x
13.1.17	LIST_ITEMS		x	o	x	x	o	x	x
13.1.18	LOCAL_PARAMETERS		x	o	x	x	o	x	x
13.1.19	MENUS		x	o	x	x	o	x	x
13.1.20	MENU_ITEMS		x	o	x	x	o	x	x
13.1.21	METHODS		x	o	x	x	o	x	x
13.1.22	METHOD_ITEMS		x	o	x	x	o	x	x
13.1.23	PARAMETERS		x	m	x	x	m	x	x
13.1.24	PARAMETER_LISTS		x	o	x	x	o	x	x
13.1.25	PLUGINS		x	o	x	x	o	x	x
13.1.26	PLUGIN_ITEMS		x	o	x	x	o	x	x
13.1.27	REFRESH_ITEMS		x	o	x	x	o	x	x
13.1.28	SEMANTIC_MAP_ITEMS		x	o	x	x	o	x	x
13.1.29	SOURCE_ITEMS		x	o	x	x	o	x	x
13.1.30	UNIT_ITEMS		x	o	x	x	o	x	x
13.1.31	WAVEFORM_ITEMS		x	o	x	x	o	x	x
13.1.32	WRITE_AS_ONE_ITEMS		x	o	x	x	o	x	x

13.1.2 AXIS_ITEMS

Objet

L'attribut AXIS_ITEMS est une liste des instances d'AXIS associées à l'attribut BLOCK_A.

Structure lexicale

AXIS_ITEMS { *reference* [, *reference*]* }

L'attribut AXIS_ITEMS est spécifié dans le Tableau 125.

Tableau 125 – Attribut AXIS_ITEMS

Type de valeurs	Description
reference	Référence à une instance d'AXIS.

13.1.3 CHARACTERISTICS

Objet

L'attribut CHARACTERISTICS spécifie les caractéristiques externes d'un attribut BLOCK_A. L'attribut CHARACTERISTICS 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 un attribut 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 126.

Tableau 126 – Attribut CHARACTERISTICS

Type de valeurs	Description
reference	Référence à une instance de RECORD.

13.1.4 CHARTS

Objet

L'attribut CHARTS spécifie les attributs CHART associés à l'attribut BLOCK_A, ainsi qu'une description et une aide pour chaque attribut CHART. Seuls les attributs CHART référencés par une instance spécifique de l'attribut BLOCK_A (voir 16.20) doivent être spécifiés.

Structure lexicale

CHARTS { [*identifiant* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs CHARTS sont spécifiés dans le Tableau 127.

Tableau 127 – Attribut CHARTS

Type de valeurs	Description
identifiant	Identificateur par lequel l'attribut CHART peut être référencé.
reference	Référence à une instance de CHART.
description	Brève description de l'attribut CHART.
help	Texte d'aide pour l'attribut CHART.

13.1.5 CHART_ITEMS

Objet

L'attribut CHART_ITEMS est une liste des instances de CHART associées à l'attribut BLOCK_A.

Structure lexicale

CHART_ITEMS { *reference* [, *reference*]* }

L'attribut CHART_ITEMS est spécifié dans le Tableau 128.

Tableau 128 – Attribut CHART_ITEMS

Type de valeurs	Description
reference	Référence à une instance de CHART.

13.1.6 COLLECTION_ITEMS

Objet

L'attribut COLLECTION_ITEMS est une liste des instances de COLLECTION associées à l'attribut BLOCK_A.

Structure lexicale

COLLECTION_ITEMS { *reference* [, *reference*]* }

L'attribut COLLECTION_ITEMS est spécifié dans le Tableau 129.

Tableau 129 – Attribut COLLECTION_ITEMS

Type de valeurs	Description
reference	Référence à une instance de COLLECTION.

13.1.7 EDIT_DISPLAY_ITEMS

Objet

L'attribut EDIT_DISPLAY_ITEMS est une liste des instances d'EDIT_DISPLAY associées à l'attribut BLOCK_A.

Structure lexicale

EDIT_DISPLAY_ITEMS { *reference* [, *reference*]* }

L'attribut EDIT_DISPLAY_ITEMS est spécifié dans le Tableau 130.

Tableau 130 – Attribut EDIT_DISPLAY_ITEMS

Type de valeurs	Description
reference	Référence à une instance d'EDIT_DISPLAY.

13.1.8 FILES

Objet

L'attribut FILES spécifie les attributs FILE associés à l'attribut BLOCK_A, ainsi qu'une description et une aide pour chaque attribut FILE. Seuls les attributs FILE référencés par une instance spécifique de l'attribut BLOCK_A (voir 16.22) doivent être spécifiés.

Structure lexicale

FILES { [*identifieur* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs FILES sont spécifiés dans le Tableau 131.

Tableau 131 – Attribut FILES

Type de valeurs	Description
identifieur	Identificateur par lequel l'attribut FILE peut être référencé.
reference	Référence à une instance de FILE.
description	Brève description de l'attribut FILE.
help	Texte d'aide pour l'attribut FILE.

13.1.9 FILE_ITEMS

Objet

L'attribut FILE_ITEMS est une liste des instances de FILE associées à l'attribut BLOCK_A.

Structure lexicale

FILE_ITEMS { *reference* [, *reference*]* }

L'attribut FILE_ITEMS est spécifié dans le Tableau 132.

Tableau 132 – Attribut FILE_ITEMS

Type de valeurs	Description
reference	Référence à une instance de FILE.

13.1.10 GRAPHS

Objet

L'attribut GRAPHS spécifie les attributs GRAPH associés à un attribut BLOCK_A, ainsi qu'une description et une aide pour chaque attribut GRAPH. Seuls les attributs GRAPH référencés par une instance spécifique de l'attribut BLOCK_A (voir 16.23) doivent être spécifiés.

Structure lexicale

GRAPHS { [*identifieur* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs GRAPHS sont spécifiés dans le Tableau 133.

Tableau 133 – Attribut GRAPHS

Type de valeurs	Description
identifier	Identificateur par lequel l'attribut GRAPH peut être référencé.
reference	Référence à une instance de GRAPH.
description	Brève description de l'attribut GRAPH.
help	Texte d'aide pour l'attribut GRAPH.

13.1.11 GRAPH_ITEMS**Objet**

L'attribut GRAPH_ITEMS est une liste des instances de GRAPH associées à l'attribut BLOCK_A.

Structure lexicale

GRAPH_ITEMS { *reference* [, *reference*]* }

L'attribut GRAPH_ITEMS est spécifié dans le Tableau 134.

Tableau 134 – Attribut GRAPH_ITEMS

Type de valeurs	Description
reference	Référence à une instance de GRAPH.

13.1.12 GRIDS**Objet**

L'attribut GRIDS spécifie les attributs GRID associés à un attribut BLOCK_A, ainsi qu'une description et une aide pour chaque attribut GRID. Seuls les attributs GRID référencés par une instance spécifique de l'attribut BLOCK_A (voir 16.24) doivent être spécifiés.

Structure lexicale

GRIDS { [*identifier* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs GRIDS sont spécifiés dans le Tableau 135.

Tableau 135 – Attribut GRIDS

Type de valeurs	Description
identifier	Identificateur par lequel l'attribut GRID peut être référencé.
reference	Référence à une instance de GRID.
description	Brève description de l'attribut GRID.
help	Texte d'aide pour l'attribut GRID.

13.1.13 GRID_ITEMS**Objet**

L'attribut GRID_ITEMS est une liste des instances de GRID associées à l'attribut BLOCK_A.

Structure lexicale

GRID_ITEMS { *reference* [, *reference*]* }

L'attribut GRID_ITEMS est spécifié dans le Tableau 136.

Tableau 136 – Attribut GRID_ITEMS

Type de valeurs	Description
reference	Référence à une instance de GRID.

13.1.14 IMAGE_ITEMS

Objet

L'attribut IMAGE_ITEMS est une liste des instances d'IMAGE associées à l'attribut BLOCK_A.

Structure lexicale

IMAGE_ITEMS { *reference* [, *reference*]* }

L'attribut IMAGE_ITEMS est spécifié dans le Tableau 137.

Tableau 137 – Attribut IMAGE_ITEMS

Type de valeurs	Description
reference	Référence à une instance d'IMAGE.

13.1.15 ITEM_ARRAY_ITEMS

Objet

L'attribut ITEM_ARRAY_ITEMS spécifie les éléments de matrice de référence associés à l'attribut BLOCK_A.

Structure lexicale

ITEM_ARRAY_ITEMS { *reference* [, *reference*]* }

L'attribut ITEM_ARRAY_ITEMS est spécifié dans le Tableau 138.

Tableau 138 – Attribut ITEM_ARRAY_ITEMS

Type de valeurs	Description
reference	Référence à une instance de matrice de référence.

13.1.16 LISTS

Objet

L'attribut LISTS spécifie les attributs LIST associés à un attribut BLOCK_A, ainsi qu'une description et une aide pour chaque attribut LIST. Seuls les attributs LIST référencés par une instance spécifique de l'attribut BLOCK_A (voir 13.1) doivent être spécifiés.

Structure lexicale

LISTS { [*identifieur* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs LISTS sont spécifiés dans le Tableau 139.

Tableau 139 – Attribut LISTS

Type de valeurs	Description
identifieur	Identificateur par lequel l'attribut LIST peut être référencé.
reference	Référence à une instance de LIST.
description	Brève description de l'attribut LIST.
help	Texte d'aide pour l'attribut LIST.

13.1.17 LIST_ITEMS

Objet

L'attribut LIST_ITEMS est une liste des instances de LIST associées à l'attribut BLOCK_A.

Structure lexicale

LIST_ITEMS { *reference* [, *reference*]* }

L'attribut LIST_ITEMS est spécifié dans le Tableau 140.

Tableau 140 – Attribut LIST_ITEMS

Type de valeurs	Description
reference	Référence à une instance de LIST.

13.1.18 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 d'un attribut 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 { [*identifieur* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs LOCAL_PARAMETERS sont spécifiés dans le Tableau 145.

13.1.19 MENUS

Objet

L'attribut MENUS spécifie les attributs MENU associés à un attribut BLOCK_A, ainsi qu'une description et une aide pour chaque attribut MENU. Seuls les attributs MENU référencés par une instance spécifique de l'attribut BLOCK_A (voir 16.27) doivent être spécifiés.

Structure lexicale

MENUS { [*identifieur* , *reference* [, *description* [, *help*?]?;]+ }

Les attributs MENUS sont spécifiés dans le Tableau 141.

Tableau 141 – Attribut MENUS

Type de valeurs	Description
identifieur	Identificateur par lequel l'attribut MENU peut être référencé.
reference	Référence à une instance de MENU.
description	Brève description de l'attribut MENU.
help	Texte d'aide pour l'attribut MENU.

13.1.20 MENU_ITEMS

Objet

L'attribut MENU_ITEMS spécifie les attributs MENU associés à l'attribut BLOCK_A.

Structure lexicale

MENU_ITEMS { *reference* [, *reference*]* }

L'attribut MENU_ITEMS est spécifié dans le Tableau 142.

Tableau 142 – Attribut MENU_ITEMS

Type de valeurs	Description
reference	Référence à un attribut MENU.

13.1.21 METHODS

Objet

L'attribut METHODS spécifie les attributs METHOD associés à un attribut BLOCK_A, ainsi qu'une description et une aide pour chaque attribut METHOD. Seuls les attributs METHOD référencés par une instance spécifique de l'attribut BLOCK_A (voir 16.28) doivent être spécifiés.

Structure lexicale

METHODS { [*identifieur* , *reference* [, *description* [, *help*?]?;]+ }

Les attributs METHODS sont spécifiés dans le Tableau 143.

Tableau 143 – Attribut METHODS

Type de valeurs	Description
identifieur	Identificateur par lequel l'attribut METHOD peut être référencé.
reference	Référence à une instance de METHOD.
description	Brève description de l'attribut METHOD.
help	Texte d'aide pour l'attribut METHOD.

13.1.22 METHOD_ITEMS

Objet

L'attribut METHOD_ITEMS spécifie les attributs METHOD associés à l'attribut BLOCK_A. Il convient de répertorier tous les attributs METHOD et les échecs d'attributs METHOD pour l'attribut BLOCK_A.

Structure lexicale

METHOD_ITEMS { *reference* [, *reference*]* }

L'attribut METHOD_ITEMS est spécifié dans le Tableau 144.

Tableau 144 – Attribut METHOD_ITEMS

Type de valeurs	Description
reference	Référence à une instance de METHOD.

13.1.23 PARAMETERS

Objet

L'attribut PARAMETERS spécifie les entrées, les sorties et les valeurs contenues associées à un attribut BLOCK_A, ainsi qu'une description et une aide pour chaque valeur. 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 d'un attribut BLOCK_A correspondent à des objets de communication individuels, répertoriés dans un dictionnaire d'objets d'équipement. Les éléments PARAMETERS font référence aux définitions de communication de ces objets. Le dictionnaire d'objets s'associe par mapping aux emplacements spécifiques des données dans un équipement.

Structure lexicale

PARAMETERS { [*identifiant* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs PARAMETERS sont spécifiés dans le Tableau 145.

Tableau 145 – Attributs PARAMETERS

Type de valeurs	Description
identifiant	Identificateur de VARIABLE, VALUE_ARRAY ou RECORD. Chaque élément de la liste des attributs PARAMETERS doit avoir un identificateur à utiliser dans l'EDD pour faire référence à celle-ci.
reference	Référence à une instance de VARIABLE, VALUE_ARRAY ou RECORD.
description	Brève description pour la référence.
help	Texte d'aide pour la référence.

13.1.24 PARAMETER_LISTS

Objet

L'attribut **PARAMETER_LISTS** spécifie les attributs **VARIABLE_LIST** associés à un attribut **BLOCK_A** qui peuvent être communiqués ensemble, ainsi qu'une description et une aide facultatives pour chaque attribut **VARIABLE_LIST**. Chaque attribut **BLOCK_A** peut contenir plusieurs attributs **PARAMETER_LISTS**.

Structure lexicale

PARAMETER_LISTS { [*identifieur* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs **PARAMETER_LISTS** sont spécifiés dans le Tableau 146.

Tableau 146 – Attributs **PARAMETER_LISTS**

Type de valeurs	Description
identifieur	Identificateur de VARIABLE_LIST . Chaque élément de la liste des attributs PARAMETER_LIST doit avoir un identificateur à utiliser dans l'EDD pour faire référence à celle-ci.
reference	Référence à une instance de VARIABLE_LIST .
description	Brève description de l'élément.
help	Texte d'aide pour l'élément.

13.1.25 PLUGINS

Objet

L'attribut **PLUGINS** spécifie les attributs **PLUGIN** associés à un attribut **BLOCK_A**, ainsi qu'une description et une aide pour chaque attribut **PLUGIN**. Seuls les attributs **PLUGIN** référencés par une instance spécifique de l'attribut **BLOCK_A** (voir 16.30) doivent être spécifiés.

Structure lexicale

PLUGINS { [*identifieur* , *reference* [, *description* [, *help*]?]?;]+ }

Les attributs **PLUGINS** sont spécifiés dans le Tableau 147.

Tableau 147 – Attribut **PLUGINS**

Type de valeurs	Description
identifieur	Identificateur par lequel l'attribut PLUGIN peut être référencé.
reference	Référence à une instance de PLUGIN .
description	Brève description de l'attribut PLUGIN .
help	Texte d'aide pour l'attribut PLUGIN .

13.1.26 PLUGIN_ITEMS

Objet

L'attribut **PLUGIN_ITEMS** spécifie les attributs **PLUGIN** associés à l'attribut **BLOCK_A**.

Structure lexicale

PLUGIN_ITEMS { *reference* [, *reference*]* }

L'attribut PLUGIN_ITEMS est spécifié dans le Tableau 148.

Tableau 148 – Attribut PLUGIN_ITEMS

Type de valeurs	Description
reference	Référence à une instance de PLUGIN.

13.1.27 REFRESH_ITEMS**Objet**

L'attribut REFRESH_ITEMS spécifie les relations REFRESH associées à l'attribut BLOCK_A.

Structure lexicale

REFRESH_ITEMS { *reference* [, *reference*]* }

L'attribut REFRESH_ITEMS est spécifié dans le Tableau 149.

Tableau 149 – Attribut REFRESH_ITEMS

Type de valeurs	Description
reference	Référence à une instance de REFRESH.

13.1.28 SEMANTIC_MAP_ITEMS**Objet**

L'attribut SEMANTIC_MAP_ITEMS spécifie les relations SEMANTIC_MAP associées à l'attribut BLOCK_A.

Structure lexicale

SEMANTIC_MAP_ITEMS { *reference* [, *reference*]* }

L'attribut SEMANTIC_MAP_ITEMS est spécifié dans le Tableau 150.

Tableau 150 – Attribut SEMANTIC_MAP_ITEMS

Type de valeurs	Description
reference	Référence à une instance de l'attribut SEMANTIC_MAP.

13.1.29 SOURCE_ITEMS**Objet**

L'attribut SOURCE_ITEMS spécifie les attributs SOURCE associés à l'attribut BLOCK_A.

Structure lexicale

SOURCE_ITEMS { *reference* [, *reference*]+ }

L'attribut SOURCE_ITEMS est spécifié dans le Tableau 151.

Tableau 151 – Attribut SOURCE_ITEMS

Type de valeurs	Description
reference	Référence à une instance de SOURCE.

13.1.30 UNIT_ITEMS**Objet**

L'attribut UNIT_ITEMS spécifie les relations UNIT associées à l'attribut BLOCK_A.

Structure lexicale

UNIT_ITEMS { *reference* [, *reference*]* }

L'attribut UNIT_ITEMS est spécifié dans le Tableau 152.

Tableau 152 – Attribut UNIT_ITEMS

Type de valeurs	Description
reference	Référence à une instance d'UNIT.

13.1.31 WAVEFORM_ITEMS**Objet**

L'attribut WAVEFORM_ITEMS spécifie les attributs WAVEFORM associés à l'attribut BLOCK_A.

Structure lexicale

WAVEFORM_ITEMS { *reference* [, *reference*]* }

L'attribut WAVEFORM_ITEMS est spécifié dans le Tableau 153.

Tableau 153 – Attribut WAVEFORM_ITEMS

Type de valeurs	Description
reference	Référence à une instance de WAVEFORM.

13.1.32 WRITE_AS_ONE_ITEMS**Objet**

L'attribut WRITE_AS_ONE_ITEMS spécifie les relations WRITE_AS_ONE associées à l'attribut BLOCK_A.

Structure lexicale

WRITE_AS_ONE_ITEMS { *reference* [, *reference*]* }

L'attribut WRITE_AS_ONE_ITEMS est spécifié dans le Tableau 154.

Tableau 154 – Attribut WRITE_AS_ONE_ITEMS

Type de valeurs	Description
reference	Référence à une instance de WRITE_AS_ONE.

13.2 BLOCK_B

13.2.1 Structure générale

Objet

Les variables d'un équipement ou module sont respectivement structurées en blocs correspondant aux composants ou aux parties fonctionnelles d'un équipement. 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 équipement, la construction BLOCK_B est utilisée conjointement avec la construction COMMAND pour permettre un adressage relatif.

NOTE 1 Cette approche fondée sur BLOCK_B est optimisée pour une meilleure efficacité de la mémoire.

NOTE 2 Le stockage d'une variable dans un équipement est spécifique au fabricant et est représenté par un répertoire d'équipements. Un répertoire d'équipements contient un récapitulatif des entrées de BLOCK_B disponibles dans un équipement. De plus, le répertoire d'équipements 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'équipement

Structure lexicale

BLOCK *identifier*

```
{
  NUMBER integer-constant | integer-expression;
  TYPE [PHYSICAL | TRANSDUCER | FUNCTION];
}
```

Les attributs BLOCK_B sont spécifiés dans le Tableau 155.

Tableau 155 – Attributs BLOCK_B

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		BLOCK_B	✓	x	✓	x	x	✓	✓
13.2.2		NUMBER	m	x	m	x	x	m	m
13.2.3		TYPE	m	x	m	x	x	m	m

13.2.2 NUMBER

Objet

L'attribut NUMBER énumère chaque attribut BLOCK_B du même TYPE dans l'équipement par un nombre entier supérieur à zéro. Un équipement peut contenir plusieurs instances de BLOCK_B, qui sont identifiées avec l'attribut NUMBER.

Structure lexicale

NUMBER *integer-constant* | *integer-expression*;

Les attributs NUMBER sont spécifiés dans le Tableau 156.

Tableau 156 – Attributs NUMBER

Type de valeurs	Description
integer-constant	Numéro de l'instance de BLOCK_B du même TYPE.
integer-expression	Expression qui doit être évaluée à un entier non négatif qui se situe dans la plage d'index de l'attribut NUMBER (pour la description des expressions, voir Article 18).

13.2.3 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 157.

Tableau 157 – Attributs TYPE

Valeur d'attribut	Description
FUNCTION	Bloc nommé constitué d'un ou de plusieurs paramètres d'entrée, de sortie ou contenus. Les blocs fonctionnels représentent les fonctions d'automatisation de base exécutées par une application EDD, qui est indépendante des équipements spécifiques et du réseau. Chaque bloc fonctionnel traite des paramètres d'entrée conformément à un algorithme spécifié et un ensemble interne de paramètres contenus. Ils génèrent des paramètres de sortie qui sont disponibles pour utilisation dans la même application EDD de bloc fonctionnel ou par d'autres applications de bloc fonctionnel.
PHYSICAL	Les caractéristiques spécifiques au matériel/aux ressources d'un équipement sont rendues visibles par l'intermédiaire du bloc physique. De manière similaire aux blocs transducteurs, ils isolent les blocs fonctionnels du matériel physique en contenant un ensemble de paramètres relatifs au matériel/aux ressources indépendants de la mise en œuvre.
TRANSDUCER	Les blocs transducteurs isolent les blocs fonctionnels des éléments spécifiques des équipements E/S, tels que des capteurs, des actionneurs et des commutateurs. Les blocs transducteurs contrôlent l'accès aux équipements E/S par l'intermédiaire d'une interface indépendante de l'équipement définie pour une utilisation par des blocs fonctionnels. Les blocs transducteurs remplissent des fonctions, telles que l'étalonnage et la linéarisation, sur des données E/S pour les convertir en représentation indépendante de l'équipement.

13.3 LIKE

Objet

L'attribut LIKE est utilisé pour créer une nouvelle instance d'EDD sur la base d'un élément existant. Sur certains équipements, par exemple, il peut exister plusieurs variables presque identiques. L'attribut LIKE fait une copie du type d'élément existant avec tous les attributs.

Structure lexicale

```

identifieur LIKE [item-type]? reference
[ {
  attribute-redefinition-specifier*
} ]?
  
```

Les attributs LIKE sont spécifiés dans le Tableau 158 et le Tableau 159.

Tableau 158 – Présentation de l'attribut LIKE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	LIKE		✓	✓	✓	✓	✓	✓	✓

Tableau 159 – Attributs LIKE

Utilisation	Type de valeurs	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-specifier	Spécifie si un attribut est ajouté, supprimé ou redéfini (voir Article 15). 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. Profils de communication: FF, ISA100 L'attribut item-type facultatif doit être utilisé.

13.4 IMPORT

Objet

Les constructions EDDL sont suffisantes pour décrire un équipement unique quelconque. Cependant, des mécanismes supplémentaires sont exigés pour décrire plusieurs versions d'un équipement unique ou pour décrire des parties d'un équipement 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 d'équipement peut être spécifiée en important simplement l'EDD de l'ancienne version d'équipement et en spécifiant les modifications apportées aux éléments.

NOTE Ce type d'EDD est appelé description delta, car l'EDD entière est spécifiée en fonction des modifications d'une EDD existante.

Trois types d'importations 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 seuls les éléments des types spécifiés sont importés. Si un type spécifié est importé, d'autres importations d'éléments spécifiques du même type ne sont pas admises;
- importation d'un élément spécifique.

L'identificateur d'un élément EDD importé ne doit pas être modifié. La Figure 63 représente les mécanismes d'importation EDDL.

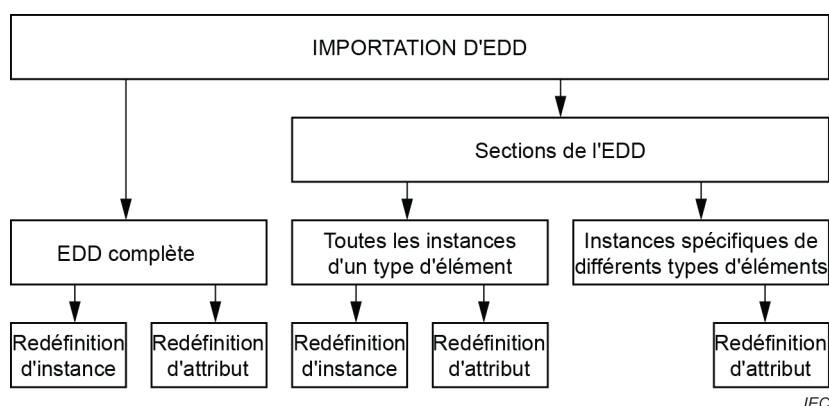


Figure 63 – Mécanismes d'importation EDDL

Une autre méthode d'importation consiste à référencer l'EDD par le nom du fichier.

Structure lexicale

Profils de communication: CS, GPE, PB, PN

```

IMPORT file-name
{
    import-keyword [& import-keyword]
    [attribute-redefinition]*
}

```

Profils de communication: CS, GPE, PB, PN

```

IMPORT
[DD_REVISION integer-constant ,
 DEVICE_REVISION integer-constant ,
 DEVICE_TYPE [integer-constant | identifieur] ,
 [EDD_VERSION integer-constant ,]?
 MANUFACTURER [integer-constant | identifieur] ,
 [MANUFACTURER_EXT string-constant]?
{
    import-keyword [& import-keyword]
    [attribute-redefinition]*
}

```

Profils de communication: FF, HART, ISA100

```

IMPORT
[DD_REVISION integer-constant ,
 DEVICE_REVISION integer-constant ,
 DEVICE_TYPE [integer-constant | identifieur] ,
 [EDD_VERSION integer-constant ,]?
 MANUFACTURER [integer-constant | identifieur] ,
{
    import-keyword [& import-keyword]
    [attribute-redefinition]*
}

```

```

import-keyword := [ EVERYTHING | AXES | BLOBS | BLOCKS_A | BLOCKS_B | CHARTS
| 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 | SEMANTIC_MAPS
| SOURCES | TEMPLATES | UNITS | VALUE_ARRAYS | VARIABLES | VARIABLE_LISTS |
WAVEFORMS | WRITE_AS_ONES | reference ]

```

Les attributs sont spécifiés dans le Tableau 160 et le Tableau 161.

Tableau 160 – Présentation de l'attribut IMPORT

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	IMPORT		✓	✓	✓	✓	✓	✓	✓

Tableau 161 – Importation de description d'un équipement

Attribut	Description
attribute-redefinition	Voir Article 15.
AXES	Toutes les instances d'AXIS d'une EDD externe sont importées.
BLOBS	Toutes les instances de BLOB d'une EDD externe sont importées.
BLOCKS_A	Toutes les instances de BLOCK_A d'une EDD externe sont importées.
BLOCKS_B	Toutes les instances de BLOCK_B d'une EDD externe sont importées.
CHARTS	Toutes les instances de CHART d'une EDD externe sont importées.
COLLECTIONS	Toutes les instances de COLLECTION d'une EDD externe sont importées.
COMMANDS	Toutes les instances de COMMAND d'une EDD externe sont importées.
COMPONENTS	Toutes les instances de COMPONENT d'une EDD externe sont importées.
COMPONENT_FOLDERS	Toutes les instances de COMPONENT_FOLDER d'une EDD externe sont importées.
COMPONENT_REFERENCES	Toutes les instances de COMPONENT_REFERENCE d'une EDD externe sont importées.
COMPONENT_RELATIONS	Toutes les instances de COMPONENT_RELATION d'une EDD externe sont importées.
DD_REVISION	Voir 8.2.
DELETE	Une instance importée doit être supprimée.
DEVICE_REVISION	Voir 8.3.
DEVICE_TYPE	Voir 8.4.
EDD_PROFILE	Voir 8.5.
EDD_VERSION	Voir 8.6.
EDIT_DISPLAYS	Toutes les instances d'EDIT_DISPLAY d'une EDD externe sont importées.
EVERYTHING	Tous les éléments d'une EDD externe sont importés. Si cette forme est utilisée, aucun type d'élément ni aucune instance unique ne peuvent être sélectionnés. Les attributs DELETE, REDEFINE et attribute-redefinition peuvent être utilisés.
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.
FILES	Toutes les instances de FILE d'une EDD externe sont importées.
GRAPHS	Toutes les instances de GRAPH d'une EDD externe sont importées.
GRIDS	Toutes les instances de GRID d'une EDD externe sont importées.
IMAGES	Toutes les instances d'IMAGE d'une EDD externe sont importées.
INTERFACES	Toutes les instances d'INTERFACE d'une EDD externe sont importées.
LISTS	Toutes les instances de LIST d'une EDD externe sont importées.
MANUFACTURER	Voir 8.8.
MANUFACTURER_EXT	Voir 8.9.

Attribut	Description
MENUS	Toutes les instances de MENU d'une EDD externe sont importées.
METHODS	Toutes les instances de METHOD d'une EDD externe sont importées.
PLUGINS	Toutes les instances de PLUGIN d'une EDD externe sont importées.
RECORDS	Toutes les instances de RECORD d'une EDD externe sont importées.
REDEFINE	Tous les attributs d'une instance importée doivent être redéfinis.
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.
REFERENCE_ARRAYS	Toutes les instances de REFERENCE_ARRAY d'une EDD externe sont importées.
REFRESHES	Toutes les instances de REFRESH d'une EDD externe sont importées.
RELATIONS	Toutes les instances des attributs REFRESH, SEMANTIC_MAP, UNIT et WRITE_AS_ONE d'une EDD externe sont importées.
RESPONSE_CODES	Toutes les instances de RESPONSE_CODES d'une EDD externe sont importées.
SEMANTIC_MAPS	Toutes les instances de l'attribut SEMANTIC_MAP d'une EDD externe sont importées.
SOURCES	Toutes les instances de SOURCE d'une EDD externe sont importées.
TEMPLATES	Toutes les instances de TEMPLATE d'une EDD externe sont importées.
UNITS	Toutes les instances d'UNIT d'une EDD externe sont importées.
VALUE_ARRAYS	Toutes les instances de VALUE_ARRAY d'une EDD externe sont importées.
VARIABLE_LISTS	Toutes les instances de VARIABLE_LIST d'une EDD externe sont importées.
VARIABLES	Toutes les instances de VARIABLE d'une EDD externe sont importées.
WAVEFORMS	Toutes les instances de WAVEFORM d'une EDD externe sont importées.
WRITE_AS_ONES	Toutes les instances de WRITE_AS_ONE d'une EDD externe sont importées.

13.5 TEMPLATE

13.5.1 Structure générale

Objet

L'attribut TEMPLATE décrit un ensemble de valeurs de paramètres utilisé pour la configuration hors ligne par l'application EDD afin de simplifier le déploiement initial des équipements. Par exemple, un équipement à pression différentielle peut être configuré comme un capteur de pression différentielle pour un échangeur de chaleur ou comme un capteur de niveau dans une application de niveau hydrostatique. Il peut y avoir plusieurs attributs TEMPLATE pour un équipement donné. L'attribut DEFAULT_VALUES définit les valeurs par défaut des paramètres.

En configuration hors ligne, l'application d'un attribut TEMPLATE modifie immédiatement les valeurs de l'ensemble de données hors ligne de sorte à équivaloir à l'attribut DEFAULT_VALUE défini dans l'attribut TEMPLATE. Cela doit être suivi par l'évaluation de toutes les conditions et de l'attribut VALIDITY.

Lorsqu'un attribut TEMPLATE est appliqué et que l'application EDD est en ligne, les valeurs de modification en cache sont mise à jour et toute valeur divergeant des valeurs d'équipement en cache est mise en évidence comme normale. Cela permet de vérifier le résultat de l'application de l'attribut TEMPLATE avant que l'utilisateur n'apporte de modifications à l'équipement ou, hors ligne, à la base de données de configuration de l'application EDD.

Structure lexicale

```
TEMPLATE identifiant
{
  LABEL string-value;
  HELP string-value;
  DEFAULT_VALUES { default-values-specifier }
  VALIDITY boolean-specifier;
}
```

Les attributs TEMPLATE sont spécifiés dans le Tableau 162.

Tableau 162 – Attributs TEMPLATE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	TEMPLATE		✓	✓	✓	✓	✓	✓	✓
13.5.2	DEFAULT_VALUES		m	m	m	m	m	m	m
14.8	HELP		o	o	o	o	o	o	o
14.9	LABEL		m	m	m	m	m	m	m
14.17	VALIDITY		o	o	o	o	o	o	o

Utilisation

La Figure 64 représente un exemple d'attribut TEMPLATE.

```
// Modèle pour la pression différentielle
TEMPLATE diff_pressure_template
{
    HELP [diff_pressure_help];
    LABEL "Differential pressure";
    DEFAULT_VALUES
    {
        upper_range = 20.0;
        lower_range = -20.0;
        myArray[0] = 10;
    }
}
```

Figure 64 – Exemples d'utilisation de l'attribut TEMPLATE

13.5.2 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 { default-values-spezifier }
default-values-spezifier :=
[reference = [expression | [expression [, expression]+]; ]+
```

Les attributs DEFAULT_VALUES sont spécifiés dans le Tableau 163.

Tableau 163 – Attribut DEFAULT_VALUES

Type de valeurs	Description
reference	Référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY, une instance de VALUE_ARRAY ou une instance de LIST.
expression	Pour un attribut VARIABLE, constante ou expression qui est évaluée à une constante spécifiant la valeur. Le type dépend du type de données spécifié avec l'attribut de référence. Pour un attribut VALUE_ARRAY ou LIST, ensemble de valeurs. Peut être imbriqué pour un attribut LIST ou VALUE_ARRAY multidimensionnel. Un attribut LIST doit être ajusté de sorte à avoir un attribut COUNT équivalent au nombre de valeurs dans l'ensemble.

Les EDD avec BLOCK_A doivent référencer les éléments de données utilisant le référencement par blocs (voir Article 16).

13.6 COMPONENT

13.6.1 Structure générale

Objet

Décrit le comportement de configuration d'un équipement 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 équipement 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'attributs OBJECT_REFERENCE. Les attributs OBJECT_REFERENCE permettent d'accéder aux variables et d'appeler des méthodes.

Structure lexicale

COMPONENT *identifier*

```
{
  LABEL string-value;
  HELP string-value;
  BYTE_ORDER [BIG_ENDIAN | LITTLE_ENDIAN];
  CAN_DELETE [FALSE | TRUE]<cond>;
  CHECK_CONFIGURATION reference;
  CLASSIFICATION classification-specifier;
  COMPONENT_PARENT reference;
  COMPONENT_PATH string-constant;
  COMPONENT_RELATIONS { reference<cond> [, reference<cond>]* }
  CONNECTION_POINT reference;
  DETECT reference;
  EDD string-constant;
  INITIAL_VALUES { [initial-value-specifier]+ }
  PRODUCT_URI string-constant;
  PROTOCOL protocol-specifier;
  REDUNDANCY integer<cond>;
  SCAN reference;
  SCAN_LIST reference;
  SUPPLIED_INTERFACE { reference [, reference]* }
}
```

Les attributs sont spécifiés dans le Tableau 164.

Tableau 164 – Attributs COMPONENT

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT	✓	x	✓	x	x	✓	✓
13.6.2		BYTE_ORDER	o	x	o	x	x	o	o
13.6.3		CAN_DELETE	o	x	o	x	x	o	o
13.6.4		CHECK_CONFIGURATION	o	x	o	x	x	o	o
14.1		CLASSIFICATION	o	x	o	x	x	o	o
14.2		COMPONENT_PARENT	o	x	o	x	x	o	o
14.3		COMPONENT_PATH	o	x	o	x	x	o	o
13.6.5		COMPONENT_RELATIONS	o	x	o	x	x	o	o
13.6.6		CONNECTION_POINT	o	x	o	x	x	o	o
13.6.7		DETECT	o	x	o	x	x	o	o
13.6.8		EDD	o	x	o	x	x	o	o
14.8		HELP	o	x	o	x	x	o	o
13.6.9		INITIAL_VALUES	o	x	o	x	x	o	o
14.9		LABEL	m	x	m	x	x	m	m
13.6.13		PRODUCT_URI	o	x	x	x	x	o	o
14.14		PROTOCOL	o	x	o	x	x	o	o
13.6.10		REDUNDANCY	o	x	o	x	x	o	o
13.6.11		SCAN	o	x	o	x	x	o	o
13.6.12		SCAN_LIST	o	x	o	x	x	o	o
14.16		SUPPLIED_INTERFACE	x	x	x	x	x	x	x

13.6.2 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 165.

Tableau 165 – Attribut BYTE_ORDER

Valeur d'attribut	Description
BIG_ENDIAN	Les octets de poids fort sont transmis avant les octets de poids faible.
LITTLE_ENDIAN	Les octets de poids faible sont transmis avant les octets de poids fort.

13.6.3 CAN_DELETE

Objet

L'attribut CAN_DELETE est égal à TRUE ou FALSE et indique si la suppression de l'instance du module en question d'un équipement modulaire est sécurisée. Cet attribut est utilisé dans des cas d'utilisation en ligne et hors ligne. Par défaut, CAN_DELETE est TRUE. Par 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] <cond>;

Les attributs CAN_DELETE sont spécifiés dans le Tableau 166.

Tableau 166 – Attribut CAN_DELETE

Valeur d'attribut	Description
TRUE	Le composant peut être supprimé.
FALSE	Le composant ne peut pas être supprimé.

13.6.4 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 doit avoir 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 *reference*;

L'attribut CHECK_CONFIGURATION est spécifié dans le Tableau 167.

Tableau 167 – Attribut CHECK_CONFIGURATION

Type de valeurs	Description
reference	Référence à un attribut METHOD.

13.6.5 COMPONENT_RELATIONS

Objet

L'attribut COMPONENT_RELATIONS répertorie les relations avec d'autres composants que l'équipement modulaire ou le module peut prendre en charge. Cette EDD peut répertorier des modules apparentés, des modules enfants et des modules parents.

Structure lexicale

COMPONENT_RELATIONS { *reference*<cond> [, *reference*<cond>]* }

L'attribut COMPONENT_RELATIONS est spécifié dans le Tableau 168.

Tableau 168 – Attribut COMPONENT_RELATIONS

Type de valeurs	Description
reference	Référence à un attribut COMPONENT_RELATION. Des expressions conditionnelles sont admises.

13.6.6 CONNECTION_POINT

Objet

L'attribut CONNECTION_POINT spécifie une référence à un attribut COLLECTION qui décrit les attributs d'un point de connexion. L'attribut COLLECTION doit faire référence aux définitions de VARIABLE qui représentent les attributs d'un point de connexion spécifié du modèle d'information.

Structure lexicale

CONNECTION_POINT *reference*;

L'attribut CONNECTION_POINT est spécifié dans le Tableau 169.

Tableau 169 – Attribut CONNECTION_POINT

Type de valeurs	Description
reference	Référence à une instance de COLLECTION.

13.6.7 DETECT

Objet

L'attribut DETECT indique la méthode qu'il convient que l'application EDD appelle pour détecter un composant.

Structure lexicale

DETECT *reference*;

L'attribut DETECT est spécifié dans le Tableau 170.

Tableau 170 – Attribut DETECT

Type de valeurs	Description
reference	Référence à une méthode de détection.

13.6.8 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-literal*;

L'attribut EDD est spécifié dans le Tableau 171.

Tableau 171 – Attribut EDD

Type de valeurs	Description
string-literal	Fichier contenant le reste de l'EDD.

13.6.9 INITIAL_VALUES

Objet

L'attribut INITIAL_VALUES est un ensemble de valeurs d'initialisation pour les attributs VARIABLE, VALUE_ARRAY ou LIST. Même si les attributs VARIABLE ont un attribut INITIAL_VALUE ou DEFAULT_VALUE, l'attribut INITIAL_VALUES de l'attribut COMPONENT doit être utilisé pour l'initialisation.

Il peut être utilisé, par exemple, pour définir différentes initialisations pour différentes utilisations, telles que la mesure de la pression différentielle, la mesure du niveau et la mesure du débit pour un transmetteur de pression.

Structure lexicale

INITIAL_VALUES { [*initial-value-specifier*]⁺ }
initial-value-specifier := [*reference* = [*expression*<cond> | *array-initialization*]]
array-initialization := { *expression*<cond> [, *expression*<cond>]^{*} }

Les attributs INITIAL_VALUES sont spécifiés dans le Tableau 172.

Tableau 172 – Attribut INITIAL_VALUES

Type de valeurs	Description
reference	Référence à un attribut VARIABLE, VALUE_ARRAY ou LIST.
expression	Valeur pour initialiser l'attribut VARIABLE référencé. Pour un attribut VALUE_ARRAY ou LIST, ensemble de valeurs. Peut être imbriqué pour un attribut LIST ou VALUE_ARRAY multidimensionnel.

13.6.10 REDUNDANCY

Objet

L'attribut REDUNDANCY correspond à un entier (par défaut égal à 1) qui indique combien de connecteurs redondants sont pris en charge par l'équipement en question.

Structure lexicale

REDUNDANCY *integer*<cond>;

L'attribut REDUNDANCY est spécifié dans le Tableau 173.

Tableau 173 – Attribut REDUNDANCY

Type de valeurs	Description
integer	Nombre de points de connexion.

13.6.11 SCAN**Objet**

L'attribut SCAN indique la méthode qu'il convient que l'application EDD appelle pour créer une liste d'actions d'un équipement modulaire.

Structure lexicale

SCAN *reference*;

L'attribut SCAN est spécifié dans le Tableau 174.

Tableau 174 – Attribut SCAN

Type de valeurs	Description
reference	Référence à la méthode d'analyse.

13.6.12 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.

Structure lexicale

SCAN_LIST *reference*;

L'attribut SCAN_LIST est spécifié dans le Tableau 175.

Tableau 175 – Attribut SCAN_LIST

Type de valeurs	Description
reference	Référence à la liste d'analyse.

13.6.13 PRODUCT_URI**Objet**

L'attribut PRODUCT_URI est un attribut de chaîne. La chaîne est une référence à une URI de produit CS 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-constant*;

L'attribut PRODUCT_URI est spécifié dans le Tableau 176.

Tableau 176 – Attribut PRODUCT_URI

Type de valeurs	Description
string-constant	Littéral de chaîne qui sert de référence à une URI de produit CS.

13.7 COMPONENT_FOLDER

Objet

Un attribut COMPONENT_FOLDER décrit un dossier dans un catalogue. Les dossiers peuvent comprendre d'autres dossiers. Les dossiers permettent de regrouper des composants, par exemple les composants d'une famille de produits. Les attributs LABEL des dossiers sont associés à l'attribut COMPONENT_PATH.

Structure lexicale

```
COMPONENT_FOLDER identifier
{
    LABEL string-value;
    HELP string-value;
    CLASSIFICATION classification-specifier;
    COMPONENT_PARENT reference;
    COMPONENT_PATH string-constant;
    PROTOCOL protocol-specifier;
}
```

Les attributs COMPONENT_REFERENCE sont spécifiés dans le Tableau 177.

Tableau 177 – Attributs COMPONENT_FOLDER

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT_FOLDER	✓	x	✓	x	x	✓	✓
14.1		CLASSIFICATION	o	x	o	x	x	o	o
14.2		COMPONENT_PARENT	o	x	o	x	x	o	o
14.3		COMPONENT_PATH	o	x	o	x	x	o	o
14.8		HELP	o	x	o	x	x	o	o
14.9		LABEL	m	x	m	x	x	m	m
14.14		PROTOCOL	o	x	o	x	x	o	o

13.8 COMPONENT_REFERENCE

Objet

Un attribut COMPONENT_REFERENCE fait référence à un autre composant par le biais de 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 équipement unique. Sinon, une famille d'équipements est indiquée.

A la place de l'attribut CLASSIFICATION, il est possible de référencer un composant avec les attributs DEVICE_TYPE et DEVICE_REVISION.

L'attribut MANUFACTURER est par défaut défini sur l'attribut MANUFACTURER de l'EDD.

Structure lexicale

```
COMPONENT_REFERENCE identifieur
{
    CLASSIFICATION classification-spezifier;
    COMPONENT_PARENT reference;
    COMPONENT_PATH string-constant;
    DEVICE_REVISION integer-constant;
    DEVICE_TYPE [integer-constant | identifieur];
    MANUFACTURER [integer-constant | identifieur];
    PROTOCOL protocol-spezifier;
}
```

Les attributs COMPONENT_REFERENCE sont spécifiés dans le Tableau 178.

Tableau 178 – Attributs COMPONENT_REFERENCE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT_REFERENCE	✓	x	✓	x	x	✓	✓
14.1		CLASSIFICATION	o	x	o	x	x	o	o
14.2		COMPONENT_PARENT	o	x	o	x	x	o	o
14.3		COMPONENT_PATH	o	x	o	x	x	o	o
8.3		DEVICE_REVISION	o	x	o	x	x	o	o
8.4		DEVICE_TYPE	o	x	o	x	x	o	o
8.8		MANUFACTURER	o	x	o	x	x	o	o
14.14		PROTOCOL	o	x	o	x	x	o	o

13.9 COMPONENT_RELATION

13.9.1 Structure générale

Objet

Un attribut 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. Il définit les attributs INTERFACE exigés et fournis pour un type spécial de relation composant-sous-composant.

La spécification de l'attribut COMPONENT peut être omise si l'attribut COMPONENT enfant décrit la relation avec un attribut PARENT_RELATION adapté.

Structure lexicale

```
COMPONENT_RELATION identifieur
{
    LABEL string-value;
    HELP string-value;
    ADDRESSING { reference [, reference]* }
    COMPONENTS { component-spezifier [, component-spezifier]* }
    MAXIMUM_NUMBER integer<cond>;
    MINIMUM_NUMBER integer<cond>;
    RELATION_TYPE relation-spezifier;
    REQUIRED_INTERFACE { reference [, reference]* }
    SUPPLIED_INTERFACE { reference [, reference]* }
}
```

Les attributs COMPONENT_RELATION sont spécifiés dans le Tableau 179.

Tableau 179 – Attributs COMPONENT_RELATION

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
		COMPONENT_RELATION	✓	x	✓	x	x	✓	✓
13.9.4		ADDRESSING	o	x	o	x	x	o	o
13.9.2		COMPONENTS	m	x	m	x	x	m	m
14.8		HELP	o	x	o	x	x	o	o
14.9		LABEL	m	x	m	x	x	m	m
13.9.5		MAXIMUM_NUMBER	o	x	o	x	x	o	o
13.9.6		MINIMUM_NUMBER	o	x	o	x	x	o	o
13.9.3		RELATION_TYPE	m	x	m	x	x	m	m
13.9.7		REQUIRED_INTERFACE	x	x	x	x	x	x	x
14.16		SUPPLIED_INTERFACE	x	x	x	x	x	x	x

13.9.2 COMPONENTS

Objet

L'attribut COMPONENTS répertorie les modules ou les équipements modulaires admis et les contraintes liées à leur utilisation.

Structure lexicale

```

COMPONENTS { component-specifier [, component-specifier]* }
component-specifier:= [ component-reference [
{ [ AUTO_CREATE integer-expression<cond> ]?
  [ FILTER boolean-expression<cond> ]?
  [ REQUIRED_RANGES { required-range-specifier
                        [ ; required-range-specifier* ] } ]?
  [ MAXIMUM_NUMBER integer-expression<cond> ]?
  [ MINIMUM_NUMBER integer-expression<cond> ]?
} ]? ]<cond>
required-range-specifier:= variable-reference
{ [ MAX_VALUE [ n ]? expression<cond> ]? [ MIN_VALUE [ n ]? expression<cond> ]? }

```

Les attributs COMPONENTS sont spécifiés dans le Tableau 180.

Tableau 180 – Attribut COMPONENTS

Attribut	Description
component-reference	COMPONENT_FOLDER ou COMPONENT dans la relation.
variable-reference	Référence à un attribut VARIABLE.
AUTO_CREATE	Nombre de composants automatiquement créés.
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.
MAXIMUM_NUMBER	Nombre maximal admis de ce composant.
MINIMUM_NUMBER	Nombre minimal exigé de ce composant.
REQUIRED_RANGES	Variables avec plages. Dans une configuration valide, il est nécessaire que les variables se situent dans la plage spécifiée. Cet attribut est généralement utilisé pour définir des restrictions pour l'adressage (par exemple, un emplacement).

Utilisation

L'exemple de la Figure 65 représente un module pouvant uniquement être branché sur les "emplacements" 0 à 4. Un "emplacement" est une variable du "GasAnalyzer_module1" qui est également "used-140" pour l'adressage. Le "module2" peut être branché sur n'importe quel emplacement.

```
COMPONENT_RELATION GasAnalyzer_module_relation
{
    RELATION_TYPE      CHILD_COMPONENT;
    ADDRESSING         { slot }

    COMPONENTS         {
        GasAnalyzer_module1 {REQUIRED_RANGES {slot{MIN_VALUE 0; MAX_VALUE 4;}}},
        GasAnalyzer_module2
    }

    MINIMUM_NUMBER     0;
    MAXIMUM_NUMBER     10;
}
```

Figure 65 – Utilisation de l'attribut REQUIRED_RANGES

13.9.3 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 relation-specifier;
relation-specifier:= [CHILD_COMPONENT | NEXT_COMPONENT | PARENT_COMPONENT |
PREV_COMPONENT | SIBLING_COMPONENT]
```

Les attributs RELATION_TYPE sont spécifiés dans le Tableau 181.

Tableau 181 – Attribut RELATION_TYPE

Attribut	Description
CHILD_COMPONENT	Liste des composants qui sont des sous-modules admis.
PARENT_COMPONENT	Liste des composants qui sont des modules ou des équipements modulaires parents admis pour ce composant.
SIBLING_COMPONENT	Relation entre des composants apparentés qui peut exiger ou exclure d'autres composants. Exigence possible avec l'attribut MINIMUM_NUMBER > 0. Exclusion possible avec l'attribut MINIMUM_NUMBER = 0.
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.
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.

Utilisation

L'exemple de la Figure 66 indique qu'il est nécessaire que "l'emplacement" suivant d'un module soit vide en spécifiant une liste COMPONENTS vide. L'attribut COMPONENT_RELATION doit être défini dans l'EDD du module où il convient que l'emplacement suivant soit vide.

```

COMPONENT_RELATION GasAnalyzer_module_relation
{
    RELATION_TYPE      NEXT_COMPONENT;
    ADDRESSING          { slot }      // liste des variables d'adressage
    COMPONENTS          { }
}

```

Figure 66 – Utilisation de l'attribut NEXT_COMPONENT**13.9.4 ADDRESSING****Objet**

L'attribut ADDRESSING spécifie une liste de variables qui cible un composant de manière unique.

Structure lexicale

ADDRESSING { *reference* [, *reference*]* }

L'attribut ADDRESSING est spécifié dans le Tableau 182.

Tableau 182 – Attribut ADDRESSING

Attribut	Description
variable-reference	Référence à une variable.

13.9.5 MAXIMUM_NUMBER

Objet

L'attribut MAXIMUM_NUMBER spécifie le nombre de composants admis. La valeur par défaut de MAXIMUM_NUMBER est 1.

Structure lexicale

MAXIMUM_NUMBER *integer*<cond>;

L'attribut MAXIMUM_NUMBER est spécifié dans le Tableau 183.

Tableau 183 – Attribut MAXIMUM_NUMBER

Type de valeurs	Description
integer	Nombre total maximal admis de l'ensemble des composants répertoriés.

13.9.6 MINIMUM_NUMBER

Objet

L'attribut MINIMUM_NUMBER spécifie le nombre de composants exigés. La valeur par défaut de MINIMUM_NUMBER est 0.

Structure lexicale

MINIMUM_NUMBER *integer*<cond>;

L'attribut MINIMUM_NUMBER est spécifié dans le Tableau 184.

Tableau 184 – Attribut MINIMUM_NUMBER

Type de valeurs	Description
integer	Nombre total minimal exigé de l'ensemble des composants répertoriés.

13.9.7 REQUIRED_INTERFACE

Objet

L'attribut REQUIRED_INTERFACE répertorie les interfaces exigées du composant associé.

Structure lexicale

REQUIRED_INTERFACE { *reference* [, *reference*]* }

L'attribut REQUIRED_INTERFACE est spécifié dans le Tableau 185.

Tableau 185 – Attribut REQUIRED_INTERFACE

Type de valeurs	Description
reference	Référence à une interface.

13.10 INTERFACE

13.10.1 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 pour un composant soient répertoriés dans l'attribut SUPPLIED_INTERFACE de l'attribut 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 EDD (attributs VARIABLE, par exemple) définies dans une interface.

Si aucune interface n'est définie pour le composant, l'accès à toutes les constructions de base est admis.

Structure lexicale

```
INTERFACE identifiant
{
    LABEL string-value;
    HELP string-value;
    DECLARATION { declaration-spezifier }
}
```

Les attributs INTERFACE sont spécifiés dans le Tableau 186.

Tableau 186 – Attributs INTERFACE

Référence	Construction de base	Attribut	Profils de communication						
			CS	FF	GPE	HART	ISA100	PB	PN
	INTERFACE		x	x	x	x	x	x	x
13.10.2	DECLARATION		x	x	x	x	x	x	x
14.8	HELP		x	x	x	x	x	x	x
14.9	LABEL		x	x	x	x	x	x	x

13.10.2 DECLARATION

Objet

L'attribut DECLARATION spécifie les constructions de base incluses dans l'interface.

Structure lexicale

```
DECLARATION { declaration-spezifier }
declaration-spezifier := [IMAGE | MENU | METHOD | VARIABLE] *
```

Les attributs DECLARATION sont spécifiés dans le Tableau 187.

Tableau 187 – Attribut DECLARATION

Attribut	Description
IMAGE	Déclarations d'IMAGE (voir 11.3).
MENU	Déclarations de MENU (voir 11.1).
METHOD	Déclarations de METHOD (voir Article 12).
VARIABLE	Déclarations de VARIABLE (voir 9.1).

14 Attributs communs

14.1 CLASSIFICATION

Objet

L'attribut CLASSIFICATION est une valeur énumérée unique qui indique les niveaux (éventuellement multiples) de classification pour cet équipement.

Structure lexicale

```
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]
```

Les attributs CLASSIFICATION sont spécifiés dans le Tableau 188.

Tableau 188 – Attributs CLASSIFICATION

Attribut
ACTUATOR
ACTUATOR_ELECTRO_PNEUMATIC
ACTUATOR_ELECTRIC
ACTUATOR_HYDRAULIC
CONVERTER
CONTROLLER
DISCRETE_IN
DISCRETE_OUT
ELECTRICAL_DISTRIBUTION
ELECTRICAL_DISTRIBUTION_POWER_MONITORING
FREQUENCY_CONVERTER

Attribut
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-constant

14.2 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 l'attribut 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.

Structure lexicale

COMPONENT_PARENT *reference;*

L'attribut COMPONENT_PARENT est spécifié dans le Tableau 189.

Tableau 189 – Attribut COMPONENT_PARENT

Type de valeurs	Description
reference	Composant parent dans le catalogue.

14.3 COMPONENT_PATH

Objet

L'attribut COMPONENT_PATH permet au fabricant 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 les attributs PROTOCOL, CLASSIFICATION et MANUFACTURER. Si un attribut COMPONENT_PARENT est utilisé, les attributs PROTOCOL, CLASSIFICATION et MANUFACTURER peuvent ne pas être définis et COMPONENT_PATH définit uniquement la sous-hiérarchie.

Structure lexicale

COMPONENT_PATH *string constant;*

L'attribut COMPONENT_PATH est spécifié dans le Tableau 190.

Tableau 190 – Attribut COMPONENT_PATH

Attribut	Description
string	Chemin d'accès spécifique au fabricant du composant dans le catalogue.

14.4 DEFINITION

Objet

L'attribut DEFINITION spécifie les actions à effectuer et est fondé sur un langage de programmation procédural (par exemple, langage de programmation C). Fondamentalement, un attribut METHOD est une fonction ou un sous-programme unique invoqué par l'application EDD lorsqu'il est choisi par l'utilisateur ou lorsqu'un événement qui survient est désigné comme devant être géré par l'attribut METHOD (par exemple, un attribut REFRESH ACTION exécuté lors de la mise à jour d'un graphique). Dans les attributs DEFINITION, des attributs METHOD et des builtins (voir l'IEC 61804-5) peuvent être invoqués. Des éléments de base EDD peuvent être utilisés dans un attribut DEFINITION.

Les éléments suivants doivent être pris en charge.

- Types de données de base (voir Article B.1).
- Matrices.
- Opérateurs (voir Article A.3).
- Instructions.
- Invocation d'attributs METHOD et de builtins.
- Attributs METHOD avec arguments (appel par valeur, appel par référence):
 - l'appel par valeur est utilisé pour transmettre une copie des données à l'attribut METHOD. Les modifications de la copie dans l'attribut METHOD ne modifient pas les données d'origine;
 - l'appel par référence est utilisé pour accéder directement aux données. Les modifications affectent les données d'origine.

Profils de communication: HART

- Les attributs METHOD avec arguments peuvent uniquement être référencés dans un autre attribut METHOD.
- DD_STRING prend en charge les opérateurs suivants: + =

Profils de communication: FF, ISA100

- Les attributs METHOD ne peuvent pas être invoqués depuis un attribut METHOD.
- Les attributs METHOD ne peuvent pas avoir d'arguments.

Structure lexicale

DEFINITION { *compound-statement* }

L'attribut DEFINITION est spécifié dans le Tableau 191.

Tableau 191 – Attribut DEFINITION

Type de valeurs	Description
compound-statement	Bloc d'instructions pour un langage de programmation procédural pris en charge.

14.5 EMPHASIS

Objet

L'attribut EMPHASIS spécifie s'il convient de mettre un élément d'interface utilisateur en évidence (un attribut WAVEFORM, par exemple) lorsqu'il est affiché. La façon dont il est mis en évidence est définie par l'application EDD. Par défaut, un élément d'interface utilisateur sans attribut EMPHASIS ne doit pas être mis en évidence.

Structure lexicale

EMPHASIS [FALSE | TRUE] <cond>;

Les attributs EMPHASIS sont spécifiés dans le Tableau 192.

Tableau 192 – Attribut EMPHASIS

Valeur d'attribut	Description
[FALSE TRUE]<cond>	Profils de communication: CS, FF, GPE, ISA100, PB, PN Il convient de mettre (TRUE) ou de ne pas mettre (FALSE) en évidence un élément d'interface utilisateur. Peut-être défini par une condition.
[FALSE TRUE]	Profils de communication: HART Il convient de mettre (TRUE) ou de ne pas mettre (FALSE) en évidence un élément d'interface utilisateur.

14.6 HANDLING

Objet

L'attribut HANDLING spécifie l'accès à la construction lorsqu'il est référencé par la construction associée à l'IU. Lorsque l'attribut HANDLING n'est pas spécifié, il est défini par défaut sur READ & WRITE, sauf pour l'attribut WAVEFORM, dont la valeur par défaut est READ.

Structure lexicale

HANDLING [READ | WRITE | READ & WRITE] <cond>;

Les attributs HANDLING sont spécifiés dans le Tableau 193.

Tableau 193 – Attribut HANDLING

Valeur d'attribut	Description
READ	La construction ne peut pas être modifiée.
WRITE	La construction peut être modifiée et écrite sur l'équipement, mais ne doit pas être initialisée depuis l'équipement.
READ & WRITE	La construction peut être modifiée, lue depuis l'équipement et écrite sur l'équipement.

14.7 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 possédait l'attribut HEIGHT XXX_SMALL. Pour l'attribut VARIABLES BIT-ENUMERATED, le comportement par défaut est d'afficher tous les bits de l'énumération. Si l'énumération comporte plus de 16 bits ou si la hauteur n'est pas suffisante pour tous les afficher, un mécanisme de défilement peut être utilisé pour pouvoir consulter tous les bits. Les attributs GRAPH, CHART et GRID sans attribut HEIGHT spécifié sont traités comme s'ils possédaient 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 194.

Tableau 194 – Attributs HEIGHT/WIDTH

Valeur d'attribut	Description
XXX_SMALL	Il convient que l'élément d'interface utilisateur occupe une partie vraiment 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 admis que pour les attributs VARIABLE.
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.
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.
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.
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.
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.
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.
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.

14.8 HELP

Objet

L'attribut HELP spécifie du texte, qui fournit 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-constant* | *string-constant*<cond>];

L'attribut HELP est spécifié dans le Tableau 195.

Tableau 195 – Attribut HELP

Type de valeurs	Description
string-constant	Profils de communication: HART, FF, ISA100 Texte lisible par l'utilisateur sans instructions conditionnelles. Les références de variable doivent être limitées à celles admises pour l'attribut string_constant décrit dans la syntaxe formelle (voir A.7.33).
string-constant<cond>	Profils de communication: CS, GPE, PB, PN Texte lisible par l'utilisateur. Peut-être défini par une condition et peut contenir tous les types de références de variable pris en charge par la syntaxe formelle.

14.9 LABEL

Objet

L'attribut LABEL spécifie la désignation affichée d'un élément.

Structure lexicale

LABEL [*string-constant* | *string-constant*<cond>];

L'attribut LABEL est spécifié dans le Tableau 196.

Tableau 196 – Attribut LABEL

Attribut	Description
string-constant	Profils de communication: FF, ISA100, HART Texte lisible par l'utilisateur sans instructions conditionnelles. Les références de variable doivent être limitées à celles admises pour l'attribut string_constant décrit dans la syntaxe formelle (voir A.7.33). Profils de communication: FF, ISA100 L'attribut LABEL de chaque attribut BLOCK_A dans l'EDD doit être unique. Si un attribut BLOCK_A est créé par la construction LIKE, l'attribut LABEL de l'attribut BLOCK_A doit être redéfini.
string-constant<cond>	Profils de communication: CS, GPE, PB, PN Texte lisible par l'utilisateur. Peut-être défini par une condition et peut contenir tous les types de références de variable pris en charge par la syntaxe formelle.

14.10 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 est affiché en utilisant une couleur par défaut définie par l'application EDD.

Structure lexicale

LINE_COLOR [*integer-expression* | *reference*<cond>]

Les couleurs sont spécifiées par l'intermédiaire d'une valeur RVB, comme défini dans la spécification W3C Cascading Style Sheets, niveau 2.

Les attributs LINE_COLOR sont spécifiés dans le Tableau 197.

Tableau 197 – Attribut LINE_COLOR

Type de valeurs	Description
integer-expression	Profils de communication: CS, FF, GPE, HART, ISA100, PB, PN Couleur à utiliser pour afficher l'élément d'interface utilisateur: entier non signé de 24 bits spécifié avec une notation hexadécimale (soit six caractères hexadécimaux). L'octet de poids fort des couleurs RVB indique l'intensité de rouge, tandis que l'octet de poids faible spécifie l'intensité de bleu.
reference<cond>	Profils de communication: CS, GPE, PB, PN Référence à une instance de VARIABLE contenant la couleur à utiliser pour afficher l'élément d'interface utilisateur. Peut-être défini par une condition.

Une liste des valeurs des couleurs RVB normalisées peut être consultée dans l'espace Standard Default Color Space for the Internet, lesquelles sont définies dans le Tableau 198. Les noms prédéfinis des couleurs normalisées du tableau ou les valeurs RVB personnalisées peuvent être utilisés dans l'attribut LINE_COLOR. Il convient de ne pas utiliser ces noms de couleurs à d'autres fins ou objectifs.

Tableau 198 – Valeurs des couleurs RVB normalisées

	Couleur	Code		Couleur	Code
	NOIR	0x000000		VERT	0x008000
	ARGENT	0XC0C0C0		VERT CITRON	0x00FF00
	GRIS	0x808080		VERT OLIVE	0x808000
	BLANC	0xFFFFFFFF		JAUNE	0xFFFF00
	MARRON	0x800000		BLEU MARINE	0x000080
	ROUGE	0xFF0000		BLEU	0x0000FF
	ORANGE	0xFFA500		BLEU SARCELLE	0x008080
	VIOLET	0x800080		TURQUOISE	0x00FFFF
	FUCHSIA	0xFF00FF			

14.11 LINE_TYPE

Objet

L'attribut LINE_TYPE est utilisé pour différencier les types de données par couleur de ligne, motif de ligne, 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 l'attribut 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

Profils de communication: CS, FF, GPE, ISA100, PB, PN

```
LINE_TYPE [DATA | DATAn | LOW_LOW_LIMIT | LOW_LIMIT | HIGH_LIMIT |
HIGH_HIGH_LIMIT | TRANSPARENT] | [DATA | DATAn | LOW_LOW_LIMIT |
LOW_LIMIT | HIGH_LIMIT | HIGH_HIGH_LIMIT | TRANSPARENT] <cond>;
```

Profils de communication: HART

```
LINE_TYPE [DATA | DATAn | LOW_LOW_LIMIT | LOW_LIMIT | HIGH_LIMIT |
HIGH_HIGH_LIMIT | TRANSPARENT] | [DATA | DATAn | LOW_LOW_LIMIT |
LOW_LIMIT | HIGH_LIMIT | HIGH_HIGH_LIMIT | TRANSPARENT] ;
```

Les attributs LINE_TYPE sont spécifiés dans le Tableau 199.

Tableau 199 – Attribut LINE_TYPE

Type de valeurs	Description
DATA	L'élément d'interface utilisateur contient des données.
n	Entier compris entre 1 et 9, qui identifie des types de lignes DATA distincts si plusieurs types sont présents.
LOW_LOW_LIMIT	L'élément d'interface utilisateur contient une limite basse inférieure.
LOW_LIMIT	L'élément d'interface utilisateur contient une limite basse.
HIGH_LIMIT	L'élément d'interface utilisateur contient une limite haute.
HIGH_HIGH_LIMIT	L'élément d'interface utilisateur contient une limite haute supérieure.
TRANSPARENT	Pour les attributs CHART et GRAPH de tendances, seuls les points de données de l'élément d'interface utilisateur sont visibles. Pour les attributs CHART d'instruments de mesure, seules les limites de la barre de données d'interface utilisateur sont visibles.

14.12 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 (identifier, reference, description, help).

L'attribut MEMBERS d'un attribut FILE ne doit pas être exprimé en utilisant une instruction conditionnelle IF, IF-ELSE, SELECT.

Structure lexicale

```
MEMBERS { members-spezifier }
members-spezifier:= [identifier , reference [, description]? [, help]?;]<cond>]*
```

Profils de communication: HART

Structure lexicale de l'attribut SOURCE MEMBERS

```
MEMBERS { members-spezifier }
members-spezifier:= identifier , reference [, description]? [,
help]?;
```

Les attributs MEMBERS sont spécifiés dans le Tableau 200.

Tableau 200 – Attributs MEMBERS

Type de valeurs	Description
identifier	Nom par lequel l'élément peut être référencé. Un nom d'identificateur spécifique peut être réutilisé dans l'attribut MEMBERS de plusieurs éléments dans la mesure où les éléments sont du même type.
reference	Référence à un élément EDD. Le type de la référence dépend des constructions de base: • CHART: référence à une instance de SOURCE; • COLLECTION: Profils de communication: CS, GPE, PB, PN, FF, ISA100 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; Profils de communication: HART Référence à un élément EDD; • FILE ou LIST: référence à une instance de COLLECTION, LIST, RECORD, REFERENCE_ARRAY, VALUE_ARRAY ou VARIABLE. Pour COLLECTION, le type de l'attribut 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.
description	Brève description de l'élément. Structure lexicale, voir 9.1.15.3.
help	Texte d'aide pour l'élément. Structure lexicale, voir 9.1.15.3.

14.13 PRIVATE

Objet

L'attribut PRIVATE spécifie si un élément est exposé hors de l'application EDD (par exemple, les applications EDD peuvent prendre en charge une interface serveur permettant d'accéder à l'équipement depuis d'autres applications). 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 201.

Tableau 201 – Attributs PRIVATE

Valeur d'attribut	Description
FALSE	L'élément est public. La valeur par défaut de PRIVATE est FALSE.
TRUE	L'élément est privé.

14.14 PROTOCOL

Objet

L'attribut PROTOCOL indique le protocole de communication utilisé pour échanger avec l'équipement modélisé par l'EDD en question. L'attribut PROTOCOL est utilisé pour séparer les EDD dans le catalogue.

Structure lexicale

PROTOCOL *protocol-specifier*;
protocol-specifier := [PROFIBUS_DP | PROFIBUS_PA | PROFINET | FF | HART | ISA100_WCI | *string-constant*]

Les attributs PROTOCOL sont spécifiés dans le Tableau 202.

Tableau 202 – Attribut PROTOCOL

Valeur d'attribut	Description
FF	Le composant est un composant FOUNDATION Fieldbus.
HART	Le composant est un composant HART.
ISA100_WCI	Le composant est un composant ISA100 WCI
PROFIBUS_DP	Le composant est un composant PROFIBUS DP.
PROFIBUS_PA	Le composant est un composant PROFIBUS PA.
PROFINET	Le composant est un composant PROFINET.
string	Spécifie un protocole quelconque. Le protocole doit être pris en charge par l'application EDD utilisée.

14.15 RESPONSE_CODES

Objet

L'attribut RESPONSE_CODES spécifie les valeurs qu'un équipement peut retourner en tant qu'information d'erreur. Chaque attribut VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST ou COMMAND peut avoir son propre ensemble d'attributs 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 *referenced-response-codes-specifier*;
referenced-response-codes-specifier := *reference*<cond>

L'attribut RESPONSE_CODES référencé est spécifié dans le Tableau 203.

Tableau 203 – Attribut RESPONSE_CODES

Attribut	Description
reference	Référence à une instance de RESPONSE_CODES. Peut-être défini par une condition.

Structure lexicale pour les attributs RESPONSE_CODES imbriqués

```
RESPONSE_CODES { response-codes-specifier }
response-codes-specifier:= [[integer , response-code-type , description ,
[help?]<cond>;]+
response-code-type:= [DATA_ENTRY_ERROR | DATA_ENTRY_WARNING | MISC_ERROR |
MISC_WARNING | MODE_ERROR | PROCESS_ERROR | SUCCESS]
```

Les attributs RESPONSE_CODES imbriqués sont spécifiés dans le Tableau 76.

14.16 SUPPLIED_INTERFACE

Objet

L'attribut SUPPLIED_INTERFACE répertorie toutes les interfaces prises en charge par ce composant.

Structure lexicale

```
SUPPLIED_INTERFACE { reference [, reference]* }
```

L'attribut SUPPLIED_INTERFACE est spécifié dans le Tableau 204.

Tableau 204 – Attribut SUPPLIED_INTERFACE

Attribut	Description
reference	Référence à une instance de l'attribut INTERFACE.

14.17 VALIDITY

Objet

Certains équipements peuvent fonctionner dans différents modes. Par exemple, un transmetteur de pression peut être capable de calculer le débit ou le niveau, mais pas les deux simultanément. Par conséquent, les données relatives au niveau peuvent ne pas être disponibles lorsque le débit est mesuré. L'attribut VALIDITY spécifie si un élément est ou n'est pas valide. Un élément sans attribut VALIDITY est toujours valide.

L'attribut VALIDITY doit être évalué avant toute utilisation de l'élément EDD. Par exemple, si l'attribut VALIDITY est FALSE, l'élément EDD n'existe pas et les valeurs ou les attributs de l'élément EDD (comme l'attribut VISIBILITY) ne sont pas pertinents.

Il convient que l'attribut 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.

Lorsque l'attribut VALIDITY d'un attribut METHOD est évalué sur FALSE, il n'a aucun effet (il n'empêche pas l'exécution) lorsque l'attribut METHOD est invoqué depuis une autre méthode ou référencé comme un attribut ACTION.

Structure lexicale

```
VALIDITY [FALSE | TRUE]<cond>;
```

Les attributs VALIDITY sont spécifiés dans le Tableau 205.

Tableau 205 – Attributs VALIDITY

Attribut	Description
FALSE	L'élément n'est pas valide.
TRUE	L'élément est valide.

14.18 VISIBILITY

Objet

L'attribut VISIBILITY spécifie si un élément est ou n'est pas visible. Un élément sans attribut VISIBILITY est toujours valide.

Il convient que l'attribut VISIBILITY soit exprimé à l'aide d'une instruction conditionnelle (IF, IF-ELSE, SELECT). L'attribut 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**] <cond>]

Les attributs VISIBILITY sont spécifiés dans le Tableau 206.

Tableau 206 – Attributs VISIBILITY

Attribut	Description
FALSE	L'élément n'est pas visible.
TRUE	L'élément est visible. La valeur par défaut de VISIBILITY est TRUE.

14.19 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 possédait l'attribut WIDTH XXX SMALL. Les attributs GRAPH, CHART et GRID sans attribut WIDTH spécifié sont traités comme s'ils possédaient l'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 194.

14.20 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'équipement.

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 207.

Tableau 207 – Attributs WRITE_MODE

Valeur d'attribut	Priorité	Description
MODE_OUT_OF_SERVICE	7 – la plus élevée	Out of Service (O/S).
MODE_INITIALIZATION	6	Initialization Manual (IMan)
MODE_LOCAL_OVERRIDE	5	Local Override (LO)
MODE_MANUAL	4	Manual (Man)
MODE_AUTOMATIC	3	Automatic (Auto)
MODE_CASCADE	2	Cascade (Cas)
MODE_REMOTE_CASCADE	1	Remote-Cascade (RCas)
MODE_REMOTE_OUTPUT	0 – la moins élevée	Remote-Output (ROut)

15 Redéfinitions

15.1 Vue d'ensemble

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'un élément de construction de base EDD.

L'attribut EVERYTHING et les directives d'importation import-item-type peuvent importer plus d'éléments EDD que ceux réellement pris en charge par l'équipement. Par conséquent, ces éléments EDD supplémentaires peuvent être supprimés dans la section REDEFINITIONS à l'aide de la syntaxe ci-dessous:

```
DELETE item-type identifier;
DELETE item-type identifier;
...
...
DELETE item-type identifier;
```

où item-type est, par exemple, VARIABLE, ARRAY, COLLECTION, etc. Outre la suppression des éléments EDD non souhaités, la liste de modification peut inclure des redéfinitions d'éléments EDD spécifiques.

Ces redéfinitions prennent la forme suivante:

```
item-type identifier { redefinitions-list }
```

Par exemple, la commande de pratique commune HART 48 peut être tronquée et est souvent redéfinie:

```
COMMAND read_additional_transmitter_status
{
    REDEFINE TRANSACTION
    {
        REQUEST {}
        REPLY
        {
            response_code,
            device_status,
            xmtr_specific_status_0,
            xmtr_specific_status_1,
            xmtr_specific_status_2,
            xmtr_specific_status_3,
            xmtr_specific_status_4,
            xmtr_specific_status_5,
            extended_device_status,
            operating_mode
        }
    }
}
```

Les relations n'ont aucun attribut et, par conséquent, peuvent uniquement être redéfinies comme un ensemble. Cela s'exprime comme suit:

REDEFINE *relation-type* *identifieur* { *relation-spezifier* }

où *relation-type* est **UNIT**, **REFRESH** ou **WRITE_AS_ONE**.

En général, chaque attribut d'élément EDD facultatif peut être supprimé ou redéfini, et les attributs obligatoires peuvent uniquement être redéfinis. Les paragraphes suivants décrivent les règles de redéfinition des éléments EDDL. La syntaxe de suppression ou de redéfinition d'un attribut est:

DELETE *attribute-type*;
REDEFINE *attribute*;

En d'autres termes, lorsqu'un attribut est supprimé, le nom de l'attribut (LABEL, HELP, VALIDITY, etc.) est inclus. Lors de la redéfinition, le mot-clé REDEFINE est utilisé suivi de la même syntaxe de déclaration d'attribut que celle habituellement utilisée pour spécifier un élément EDD. Par exemple,

```
DELETE LABEL;
DELETE HELP;
REDEFINE VALIDITY SELECT ( measurement_type )
{
    CASE 0: { TRUE; }
    CASE 9: { TRUE; }
    CASE 16: { TRUE; }
    DEFAULT: { FALSE; }
}
```

Certains attributs ont des sous-attributs ou des listes d'attributs. Les variables, les matrices de référence, les commandes et les collections sont des exemples d'éléments DD disposant de ces déclarations composées. Pour les sous-attributs, la redéfinition peut s'exprimer comme suit:

```
Attribute
{
    DELETE sub-attribute-type;
    REDEFINE sub-attribute
}
```

Par exemple,

```
GRAPH tank_level
{
    TYPE YT
    {
        DELETE NUMBER_OF_POINTS;
        REDEFINE X_INCREMENT 0.02;
    }
}
```

ou

```
VARIABLE lower_range_value
{
    TYPE FLOAT
    {
        REDEFINE DISPLAY_FORMAT ".1f";
        DELETE SCALING_FACTOR;
    }
}
```

Les attributs contenant des listes (énumérateurs, membres et codes de réponse, par exemple) peuvent ajouter (ADD), supprimer (DELETE) ou redéfinir (REDEFINE) les éléments de la liste. Ces redéfinitions s'expriment comme suit:

```
Attribute
{
    DELETE element;
    ADD element-specifier
    REDEFINE element-specifier
}
```

Par exemple,

```
VARIABLE pressure_units
{
    TYPE ENUMERATED
    {
        ADD {7, "Bar"}
        ADD {12, "kPa"}
    }
}
```

ou

```
COLLECTON pressure
{
    MEMBERS
    {
        REDEFINE DIGITAL_UNITS, pressure_range_units;
        ADD UPPER_TRIM_POINT, pressure_utp;
        ADD LOWER_TRIM_POINT, pressure_ltp;
    }
}
```

ADD, DELETE et REDEFINE sont des mots-clés utilisés pour modifier les caractéristiques d'un élément DD et par conséquent adapter le comportement afin de modifier correctement l'équipement. DELETE et REDEFINE peuvent être utilisés pour la plupart des attributs, tandis qu'ADD permet d'ajouter de nouveaux sous-attributs à une liste d'attributs.

Les attributs de redéfinition sont spécifiés dans le Tableau 208.

Tableau 208 – Attributs de redéfinition

Attribut	Description
ADD	Ajoute l'attribut spécifié suivant. "A" indique qu'ADD peut être appliqué à l'attribut dans les tableaux des règles de redéfinition (voir Tableau 209 à Tableau 238).
DELETE	Supprime l'attribut spécifié suivant. "D" indique que DELETE peut être appliqué à l'attribut dans les tableaux des règles de redéfinition (voir Tableau 209 à Tableau 238).
REDEFINE	Redéfinit l'attribut spécifié suivant. "R" indique que REDEFINE peut être appliqué à l'attribut dans les tableaux des règles de redéfinition (voir Tableau 209 à Tableau 238).

15.2 AXIS

Structure lexicale

AXIS *identifiant*

```
{
  [[DELETE | REDEFINE] [CONSTANT_UNIT | HELP | LABEL | MAX_VALUE |
    MIN_VALUE | SCALING]] *
}
```

Les règles de redéfinition pour les attributs AXIS sont spécifiées dans le Tableau 208 et le Tableau 209.

Tableau 209 – Règles de redéfinition pour les attributs AXIS

A	D	R	Attribut	Description
	•	•	CONSTANT_UNIT	Élément lexical (voir 9.1.3).
	•	•	HELP	Élément lexical (voir 14.8).
	•	•	LABEL	Élément lexical (voir 14.9).
	•	•	MAX_VALUE	Élément lexical (voir 11.7.2).
	•	•	MIN_VALUE	Élément lexical (voir 11.7.2).
	•	•	SCALING	Élément lexical (voir 11.7.3).

15.3 BLOB

Structure lexicale

BLOB *identifiant*

```
{
  [[DELETE | REDEFINE] [HANDLING | HELP | IDENTITY | LABEL]] *
}
```

Les règles de redéfinition pour les attributs BLOB sont spécifiées dans le Tableau 208 et le Tableau 210.

Tableau 210 – Règles de redéfinition pour les attributs BLOB

A	D	R	Attribut	Description
		•	HANDLING	Elément lexical (voir 14.6).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	IDENTITY	Elément lexical (voir 9.8.2).
	•	•	LABEL	Elément lexical (voir 14.9).

15.4 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 | ITEM_ARRAY_ITEMS | LIST_ITEMS | LOCAL_PARAMETERS |
  local-parameters-list-element | MENU_ITEMS | METHOD_ITEMS |
  PARAMETER_LISTS | parameter-lists-list-element | PLUGIN_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 208 et le Tableau 211.

Tableau 211 – Règles de redéfinition pour les attributs BLOCK_A

A	D	R	Attribut	Description
		•	CHARACTERISTICS	Elément lexical (voir 13.1.3).
		•	LABEL	Elément lexical (voir 14.9).
		•	PARAMETERS	Elément lexical (voir 13.1.23).
•		•	parameters-list-element	La liste des éléments individuels de l'attribut PARAMETERS peut être modifiée.
	•	•	AXIS_ITEMS	Elément lexical (voir 13.1.2).
	•	•	CHART_ITEMS	Elément lexical (voir 13.1.5).
	•	•	COLLECTION_ITEMS	Elément lexical (voir 13.1.6).
	•	•	EDIT_DISPLAY_ITEMS	Elément lexical (voir 13.1.7).
	•	•	FILE_ITEMS	Elément lexical (voir 13.1.9).
	•	•	GRAPH_ITEMS	Elément lexical (voir 13.1.11).
	•	•	GRID_ITEMS	Eléments lexicaux (voir 13.1.13).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	IMAGE_ITEMS	Eléments lexicaux (voir 13.1.14).
	•	•	ITEM_ARRAY_ITEMS	Elément lexical (voir 13.1.15).
	•	•	LIST_ITEMS	Elément lexical (voir 13.1.17).
		•	LOCAL_PARAMETERS	Elément lexical (voir 13.1.18).
•		•	local-parameters-list-element	La liste des éléments individuels de l'attribut LOCAL_PARAMETERS peut être modifiée.
	•	•	MENU_ITEMS	Elément lexical (voir 13.1.20).
	•	•	METHOD_ITEMS	Elément lexical (voir 13.1.22).
		•	PARAMETER_LISTS	Elément lexical (voir 13.1.23 et 13.1.24).
•		•	parameter-lists-list-element	La liste des éléments individuels de l'attribut PARAMETERS_LISTS peut être modifiée.
	•	•	PLUGIN_ITEMS	Elément lexical (voir 13.1.26).
	•	•	REFRESH_ITEMS	Elément lexical (voir 13.1.27).
			SEMANTIC_MAP_ITEMS	Elément lexical (voir 13.1.28).
	•	•	SOURCE_ITEMS	Elément lexical (voir 13.1.29).
	•	•	UNIT_ITEMS	Elément lexical (voir 13.1.30).
	•	•	WAVEFORM_ITEMS	Elément lexical (voir 13.1.31).
	•	•	WRITE_AS_ONE_ITEMS	Elément lexical (voir 13.1.32).
	•	•	CHARTS	Elément lexical (voir 13.1.4).
•	•	•	charts-element	Les éléments individuels de l'attribut CHARTS peuvent être modifiés.
	•	•	FILES	Elément lexical (voir 13.1.8).
•	•	•	files-element	Les éléments individuels de l'attribut FILES peuvent être modifiés.
	•	•	LISTS	Elément lexical (voir 13.1.16).
•	•	•	lists-element	Les éléments individuels de l'attribut LISTS peuvent être modifiés.
	•	•	GRAPHS	Elément lexical (voir 13.1.10).
•	•	•	graphs-element	Les éléments individuels de l'attribut GRAPHS peuvent être modifiés.
	•	•	GRIDS	Elément lexical (voir 13.1.12).
•	•	•	grids-element	Les éléments individuels de l'attribut GRIDS peuvent être modifiés.
	•	•	MENUS	Elément lexical (voir 13.1.19).
•	•	•	menus-element	Les éléments individuels de l'attribut MENUS peuvent être modifiés.
	•	•	METHODS	Elément lexical (voir 13.1.21).
•	•	•	methods-element	Les éléments individuels de l'attribut METHODS peuvent être modifiés.
	•	•	PLUGINS	Elément lexical (voir 13.1.25).
•	•	•	plugins-element	Les éléments individuels de l'attribut PLUGIN peuvent être modifiés.

15.5 BLOCK_B

Structure lexicale

```
BLOCK identifier
{
  [REDEFINE [NUMBER | TYPE]] *
}
```

Les règles de redéfinition pour les attributs BLOCK_B sont spécifiées dans le Tableau 208 et le Tableau 212.

Tableau 212 – Règles de redéfinition pour les attributs BLOCK_B

A	D	R	Attribut	Description
		•	NUMBER	Élément lexical (voir 13.2.2).
		•	TYPE	Élément lexical (voir 13.2.3).

15.6 CHART

Structure lexicale

```
CHART identifier
{
  [[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 208 et le Tableau 213.

Tableau 213 – Règles de redéfinition pour les attributs CHART

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 14.12).
•	•	•	members-list-element	La liste des éléments individuels de l'attribut MEMBERS peut être modifiée.
	•	•	CYCLE_TIME	Élément lexical (voir 11.5.2).
	•	•	HEIGHT	Élément lexical (voir 14.7).
	•	•	HELP	Élément lexical (voir 14.8).
	•	•	LABEL	Élément lexical (voir 14.9).
	•	•	LENGTH	Élément lexical (voir 11.5.3).
	•	•	TYPE	Élément lexical (voir 11.5.4).
	•	•	VALIDITY	Élément lexical (voir 14.17).
	•	•	VISIBILITY	Élément lexical (voir 14.18).
	•	•	WIDTH	Élément lexical (voir 14.19).

15.7 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 208 et le Tableau 214.

Tableau 214 – Règles de redéfinition pour les attributs COLLECTION

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 14.12).
•	•	•	members-list-element	La liste des éléments individuels de l'attribut MEMBERS peut être modifiée.
	•	•	HELP	Élément lexical (voir 14.8).
	•	•	LABEL	Élément lexical (voir 14.9).
	•	•	PRIVATE	Élément lexical (voir 14.13).
	•	•	VALIDITY	Élément lexical (voir 14.17).
	•	•	VISIBILITY	Élément lexical (voir 14.18).

15.8 COMMAND

Structure lexicale

```
COMMAND identifieur
{
  [[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 208 et le Tableau 215.

Tableau 215 – Règles de redéfinition pour les attributs COMMAND

A	D	R	Attribut	Description
			OPERATION	Élément lexical (voir 10.1.8).
•	•	•	TRANSACTION	Élément lexical (voir 10.1.11).
	•	•	INDEX	Élément lexical (voir 10.1.6).
			BLOCK_B	Élément lexical (voir 10.1.4).
			NUMBER	Élément lexical (voir 10.1.7).
	•	•	SLOT	Élément lexical (voir 10.1.9).
	•	•	SUB_SLOT	Élément lexical (voir 10.1.10).
	•	•	API	Élément lexical (voir 10.1.3).
	•	•	HEADER	Élément lexical (voir 10.1.5).
	•	•	RESPONSE_CODES	Élément lexical (voir 14.15).
•	•	•	response-codes-list-element	La liste des éléments individuels de l'attribut RESPONSE_CODES peut être modifiée.

15.9 COMPONENT

Structure lexicale

```
COMPONENT identifiant
{
  [[DELETE | REDEFINE] [LABEL | BYTE_ORDER | CAN_DELETE |
    CHECK_CONFIGURATION | CLASSIFICATION | CONNECTION_POINT | HELP |
    COMPONENT_PARENT | COMPONENT_PATH | COMPONENT_RELATIONS |
    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 208 et le Tableau 216.

Tableau 216 – Règles de redéfinition pour les attributs COMPONENT

A	D	R	Attribut	Description
		•	LABEL	Élément lexical (voir 14.9).
	•	•	BYTE_ORDER	Élément lexical (voir 13.6.2).
	•	•	CAN_DELETE	Élément lexical (voir 13.6.3).
	•	•	CHECK_CONFIGURATION	Élément lexical (voir 13.6.4).
	•	•	CLASSIFICATION	Élément lexical (voir 14.1).
	•	•	COMPONENT_PARENT	Élément lexical (voir 14.2).
	•	•	COMPONENT_PATH	Élément lexical (voir 14.3).
	•	•	COMPONENT_RELATIONS	Élément lexical (voir 13.6.5).
	•	•	CONNECTION_POINT	Élément lexical (voir 13.6.6).
	•	•	DETECT	Élément lexical (voir 13.6.7).
	•	•	EDD	Élément lexical (voir 13.6.8).
	•	•	HELP	Élément lexical (voir 14.8).
	•	•	INITIAL_VALUES	Élément lexical (voir 13.6.9).
	•	•	PRODUCT_URI	Élément lexical (voir 13.6.13).
	•	•	PROTOCOL	Élément lexical (voir 14.14).
	•	•	REDUNDANCY	Élément lexical (voir 13.6.10).
	•	•	SCAN	Élément lexical (voir 13.6.11).
	•	•	SCAN_LIST	Élément lexical (voir 13.6.12).
	•	•	SUPPLIED_INTERFACE	Élément lexical (voir 14.16).

15.10 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 208 et le Tableau 217.

Tableau 217 – Règles de redéfinition pour les attributs COMPONENT_FOLDER

A	D	R	Attribut	Description
		•	LABEL	Elément lexical (voir 14.9).
	•	•	CLASSIFICATION	Elément lexical (voir 14.1).
	•	•	COMPONENT_PARENT	Elément lexical (voir 14.2).
	•	•	COMPONENT_PATH	Elément lexical (voir 14.3).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	PROTOCOL	Elément lexical (voir 14.14).

15.11 COMPONENT_REFERENCE

Structure lexicale

COMPONENT_REFERENCE *identifier*

```
{
  [[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 208 et le Tableau 218.

Tableau 218 – Règles de redéfinition pour les attributs COMPONENT_REFERENCE

A	D	R	Attribut	Description
	•	•	CLASSIFICATION	Elément lexical (voir 14.1).
	•	•	COMPONENT_PARENT	Elément lexical (voir 14.2).
	•	•	COMPONENT_PATH	Elément lexical (voir 14.3).
	•	•	DEVICE_REVISION	Elément lexical (voir 8.3).
	•	•	DEVICE_TYPE	Elément lexical (voir 8.4).
	•	•	MANUFACTURER	Elément lexical (voir 8.8).
	•	•	PROTOCOL	Elément lexical (voir 14.14).

15.12 COMPONENT_RELATION

Structure lexicale

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] ] *
```

Les règles de redéfinition pour les attributs COMPONENT_RELATION sont spécifiées dans le Tableau 208 et le Tableau 219.

Tableau 219 – Règles de redéfinition pour les attributs COMPONENT_RELATION

A	D	R	Attribut	Description
		•	COMPONENTS	Elément lexical (voir 13.9.2).
•	•	•	composants-list-element	La liste des éléments individuels de l'attribut COMPONENTS peut être modifiée.
	•	•	AUTO_CREATE	Elément lexical (voir 13.9.2).
	•	•	FILTER	Elément lexical (voir 13.9.2).
	•	•	REQUIRED_RANGES	Elément lexical (voir 13.9.2).
		•	LABEL	Elément lexical (voir 14.9).
		•	RELATION_TYPE	Elément lexical (voir 13.9.3).
	•	•	ADDRESSING	Elément lexical (voir 13.9.4).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	MAXIMUM_NUMBER	Elément lexical (voir 13.9.5).
	•	•	MINIMUM_NUMBER	Elément lexical (voir 13.9.6).
	•	•	REQUIRED_INTERFACE	Elément lexical (voir 13.9.7).
	•	•	SUPPLIED_INTERFACE	Elément lexical (voir 14.16).

15.13 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 208 et le Tableau 220.

Tableau 220 – Règles de redéfinition pour les attributs EDIT_DISPLAY

A	D	R	Attribut	Description
		•	EDIT_ITEMS	Elément lexical (voir 11.2.2).
		•	LABEL	Elément lexical (voir 14.9).
	•	•	DISPLAY_ITEMS	Elément lexical (voir 11.2.3).
	•	•	POST_EDIT_ACTIONS	Elément lexical (voir 11.2.4).
	•	•	PRE_EDIT_ACTIONS	Elément lexical (voir 11.2.5).

15.14 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 208 et le Tableau 221.

Tableau 221 – Règles de redéfinition pour les attributs FILE

A	D	R	Attribut	Description
		•	MEMBERS	Elément lexical (voir 14.12).
•	•	•	members-list-element	La liste des éléments individuels de l'attribut MEMBERS peut être modifiée.
	•	•	HANDLING	Elément lexical (voir 14.6).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	IDENTITY	Elément lexical (voir 9.6.2).
	•	•	LABEL	Elément lexical (voir 14.9).
	•	•	SHARED	Elément lexical (voir 9.6.4).
	•	•	ON_UPDATE_ACTIONS	Elément lexical (voir 9.6.3).

15.15 GRAPH

Structure lexicale

```

GRAPH identifier
{
    [[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 208 et le Tableau 222.

Tableau 222 – Règles de redéfinition pour les attributs GRAPH

A	D	R	Attribut	Description
		•	MEMBERS	Elément lexical (voir 14.12).
•	•	•	members-list-element	La liste des éléments individuels de l'attribut MEMBERS peut être modifiée.
	•	•	CYCLE_TIME	Elément lexical (voir 11.8.2).
	•	•	HEIGHT	Elément lexical (voir 14.7).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	LABEL	Elément lexical (voir 14.9).
	•	•	VALIDITY	Elément lexical (voir 14.17).
	•	•	VISIBILITY	Elément lexical (voir 14.18).
	•	•	WIDTH	Elément lexical (voir 14.19).
	•	•	X_AXIS	Elément lexical (voir 11.8.3).

15.16 GRID

Structure lexicale

```

GRID identifier
{
    [[ADD | DELETE | REDEFINE] [VECTORS | HANDLING | HEIGHT | HELP | LABEL |
    ORIENTATION | VALIDITY | VISIBILITY | WIDTH]]*
}

```

Les règles de redéfinition pour les attributs GRAPH sont spécifiées dans le Tableau 208 et le Tableau 223.

Tableau 223 – Règles de redéfinition pour les attributs GRID

A	D	R	Attribut	Description
		•	VECTORS	Elément lexical (voir 11.4.3).
	•	•	HANDLING	Elément lexical (voir 14.6).
	•	•	HEIGHT	Elément lexical (voir 14.7).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	LABEL	Elément lexical (voir 14.9).
	•	•	ORIENTATION	Elément lexical (voir 11.4.2).
	•	•	VALIDITY	Elément lexical (voir 14.17).
	•	•	VISIBILITY	Elément lexical (voir 14.18).
	•	•	WIDTH	Elément lexical (voir 14.19).

15.17 IMAGE**Structure lexicale**

```

IMAGE identifiant
{
  [[DELETE | REDEFINE] [PATH | HELP | LABEL | LINK | VALIDITY |
    VISIBILITY]] *
}

```

Les règles de redéfinition pour les attributs GRAPH sont spécifiées dans le Tableau 208 et le Tableau 224.

Tableau 224 – Règles de redéfinition pour les attributs IMAGE

A	D	R	Attribut	Description
		•	PATH	Elément lexical (voir 11.3.3).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	LABEL	Elément lexical (voir 14.9).
	•	•	LINK	Elément lexical (voir 11.3.2).
	•	•	VALIDITY	Elément lexical (voir 14.17).
	•	•	VISIBILITY	Elément lexical (voir 14.18).

15.18 INTERFACE**Structure lexicale**

```

INTERFACE identifiant
{
  [[DELETE | REDEFINE] [DECLARATION | LABEL | HELP]] *
}

```

Les règles de redéfinition pour les attributs INTERFACE sont spécifiées dans le Tableau 208 et le Tableau 225.

Tableau 225 – Règles de redéfinition pour les attributs INTERFACE

A	D	R	Attribut	Description
		•	DECLARATION	Elément lexical (voir 13.10.2).
		•	LABEL	Elément lexical (voir 14.9).
	•	•	HELP	Elément lexical (voir 14.8).

15.19 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 208 et le Tableau 226.

Tableau 226 – Règles de redéfinition pour les attributs LIST

A	D	R	Attribut	Description
		•	TYPE	Elément lexical (voir 9.4.4).
	•	•	CAPACITY	Elément lexical (voir 9.4.2).
	•	•	COUNT	Elément lexical (voir 9.4.3).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	LABEL	Elément lexical (voir 14.9).
	•	•	PRIVATE	Elément lexical (voir 14.13).
	•	•	VALIDITY	Elément lexical (voir 14.17).
	•	•	VISIBILITY	Elément lexical (voir 14.18).

15.20 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 208 et le Tableau 227.

Tableau 227 – Règles de redéfinition pour les attributs MENU

A	D	R	Attribut	Description
		•	ITEMS	Elément lexical (voir 11.1.5).
		•	LABEL	Elément lexical (voir 14.9).
	•	•	ACCESS	Elément lexical (voir 11.1.2).
	•	•	HELP	Elément lexical (voir 14.8).
	•	•	EXIT_ACTIONS	Elément lexical (voir 11.1.3).
	•	•	INIT_ACTIONS	Elément lexical (voir 11.1.4).
	•	•	POST_EDIT_ACTIONS	Elément lexical (voir 11.1.6).
	•	•	POST_READ_ACTIONS	Elément lexical (voir 11.1.7).
	•	•	POST_WRITE_ACTIONS	Elément lexical (voir 11.1.8).
	•	•	PRE_EDIT_ACTIONS	Elément lexical (voir 11.1.9).
	•	•	PRE_READ_ACTIONS	Elément lexical (voir 11.1.10).
	•	•	PRE_WRITE_ACTIONS	Elément lexical (voir 11.1.11).
	•	•	STYLE	Elément lexical (voir 11.1.12).
	•	•	VALIDITY	Elément lexical (voir 14.17).
	•	•	VISIBILITY	Elément lexical (voir 14.18).

15.21 METHOD

Structure lexicale

METHOD *identifiant*

```
{
  [[DELETE | REDEFINE] [DEFINITION | ACCESS | CLASS | LABEL | HELP |
    PRIVATE | VALIDITY | VISIBILITY]] *
}
```

Les règles de redéfinition pour les attributs METHOD sont spécifiées dans le Tableau 208 et le Tableau 228.

Tableau 228 – Règles de redéfinition pour les attributs METHOD

A	D	R	Attribut	Description
		•	DEFINITION	Elément lexical (voir 14.4).
	•	•	ACCESS	Elément lexical (voir 12.2).
		•	CLASS	Elément lexical (voir 12.3).
	•	•	LABEL	Elément lexical (voir 14.9).
	•	•	HELP	Elément lexical (voir 14.8).
			TYPE	Elément lexical (voir 12.4). TYPE ne doit pas être redéfini.
	•	•	PRIVATE	Elément lexical (voir 14.13).
	•	•	VALIDITY	Elément lexical (voir 14.17).
	•	•	VISIBILITY	Elément lexical (voir 14.18).