

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Function blocks (FB) for process control –
Part 3: Electronic Device Description Language (EDDL)**

**Blocs fonctionnels (FB) pour les procédés industriels –
Partie 3: Langage de description électronique de produit (EDDL)**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2006 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 la CEI ou du Comité national de la CEI du pays du demandeur.

Si vous avez des questions sur le copyright de la CEI ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de la CEI de votre pays de résidence.

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

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

About the IEC

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

About IEC publications

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

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you 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It also gives information on projects, replaced and withdrawn publications.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 on-line and also once a month by email.

Electropedia - www.electropedia.org

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

Customer Service Centre - webstore.iec.ch/csc

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

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) 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 CEI

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

Liens utiles:

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

La recherche avancée vous permet de trouver des publications CEI 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.

Just Published CEI - webstore.iec.ch/justpublished

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

Electropedia - www.electropedia.org

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

Service Clients - webstore.iec.ch/csc

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

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Function blocks (FB) for process control –
Part 3: Electronic Device Description Language (EDDL)**

**Blocs fonctionnels (FB) pour les procédés industriels –
Partie 3: Langage de description électronique de produit (EDDL)**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XH

ICS 25.040.40; 35.240.50

ISBN 978-2-83220-541-9

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

CONTENTS

FOREWORD.....	16
INTRODUCTION.....	18
1 Scope.....	19
2 Normative references.....	19
3 Terms, definitions, abbreviated terms and acronyms	20
4 Conformance statement.....	26
5 Conventions for lexical structures	27
6 EDD and EDDL model	28
6.1 Overview of EDD and EDDL.....	28
6.2 EDD architecture	28
6.3 Concepts of EDD	28
6.4 Principles of the EDD development process	29
6.5 Interrelations between the lexical structure and formal definitions.....	30
6.6 Builtins	30
6.7 Profiles.....	30
7 Electronic Device Description Language.....	30
7.1 Overview	30
7.2 EDD identification information.....	40
7.3 AXIS.....	43
7.4 BLOCK.....	44
7.5 CHART.....	53
7.6 COLLECTION.....	56
7.7 COMMAND.....	57
7.8 CONNECTION.....	63
7.9 DOMAIN.....	64
7.10 EDIT_DISPLAY.....	65
7.11 FILE.....	67
7.12 GRAPH.....	67
7.13 GRID.....	69
7.14 IMAGE.....	72
7.15 IMPORT	74
7.16 LIKE.....	85
7.17 LIST	86
7.18 MENU	87
7.19 METHOD.....	98
7.20 PROGRAM	100
7.21 RECORD.....	101
7.22 REFERENCE_ARRAY	101
7.23 Relations	102
7.24 RESPONSE_CODES	103
7.25 SOURCE	104
7.26 VALUE_ARRAY	107
7.27 VARIABLE.....	108
7.28 VARIABLE_LIST.....	123

7.29 WAVEFORM	124
7.30 Common attributes	131
7.31 Output redirection (OPEN and CLOSE)	134
7.32 Conditional expression.....	134
7.33 Referencing	135
7.34 Strings.....	141
7.35 Expression	143
7.36 Text dictionary	149
Annex A (normative) EDDL formal definition	150
Annex B (normative) EDDL Builtin library	224
Annex C (informative) EDD example	337
Annex D (normative) Profiles of EDDL and Builtins	350
Annex E (informative) Historical background	376
Bibliography	377
Figure 1 – Position of the IEC 61804 series related to other standards and products	18
Figure 2 – EDD generation process	29
Figure 3 – BLOCK_A	32
Figure 4 – CHART	32
Figure 5 – COLLECTION	33
Figure 6 – COMMAND	33
Figure 7 – DOMAIN	33
Figure 8 – EDIT_DISPLAY	34
Figure 9 – FILE	34
Figure 10 – GRAPH	34
Figure 11 – GRID	34
Figure 12 – IMAGE	35
Figure 13 – LIKE	35
Figure 14 – LIST	35
Figure 15 – MENU	36
Figure 16 – PROGRAM	36
Figure 17 – RECORD	36
Figure 18 – REFERENCE_ARRAY	37
Figure 19 – REFRESH	37
Figure 20 – UNIT	37
Figure 21 – WRITE_AS_ONE	37
Figure 22 – SOURCE	38
Figure 23 – VALUE_ARRAY	38
Figure 24 – VARIABLE	38
Figure 25 – VARIABLE_LIST	38
Figure 26 – WAVEFORM.....	39
Figure 27 – EDDL import mechanisms.....	74
Figure 28 – MENU activation (ACCESS OFFLINE).....	94

Figure 29 – Action performed after a new value is entered	95
Figure 30 – Action performed after all VARIABLE inputs of the MENU are accepted (ACCESS OFFLINE)	95
Figure 31 – Method execution	95
Figure 32 – MENU activation (ACCESS ONLINE)	96
Figure 33 – Cyclic reading of dynamic VARIABLES (ACCESS ONLINE)	97
Figure 34 – Action performed after all VARIABLE inputs of the MENU are accepted (ACCESS ONLINE)	97
Figure 35 – Time for read-and-write operation	122
Figure C.1 – Example of an operator screen using EDD	337
Table 1 – Field attribute descriptions	27
Table 2 – DD_REVISION attribute	40
Table 3 – DEVICE_REVISION attribute	41
Table 4 – DEVICE_TYPE attribute	41
Table 5 – EDD_PROFILE attribute	41
Table 6 – EDD_VERSION attribute	42
Table 7 – MANUFACTURER attribute	42
Table 8 – MANUFACTURER_EXT attribute	42
Table 9 – AXIS attributes	43
Table 10 – MAX_VALUE, MIN_VALUE attribute	43
Table 11 – SCALING attribute	44
Table 12 – BLOCK_A attributes	45
Table 13 – CHARACTERISTIC attribute	45
Table 14 – PARAMETER attributes	46
Table 15 – AXIS_ITEMS attribute	46
Table 16 – CHART_ITEMS attribute	46
Table 17 – COLLECTION_ITEMS attribute	47
Table 18 – EDIT_DISPLAY_ITEMS attribute	47
Table 19 – FILE_ITEMS attribute	47
Table 20 – GRAPH_ITEMS attribute	48
Table 21 – GRID_ITEMS attribute	48
Table 22 – IMAGE_ITEMS attribute	48
Table 23 – LIST_ITEMS attribute	48
Table 24 – MENU_ITEMS attribute	49
Table 25 – METHOD_ITEMS attribute	49
Table 26 – PARAMETER_LISTS attributes	50
Table 27 – REFERENCE_ARRAY_ITEMS attribute	50
Table 28 – REFRESH_ITEMS attribute	50
Table 29 – SOURCE_ITEMS attribute	51
Table 30 – UNIT_ITEMS attribute	51
Table 31 – WAVEFORM_ITEMS attribute	51
Table 32 – WRITE_AS_ONE_ITEMS attribute	51
Table 33 – BLOCK_B attributes	52

Table 34 – NUMBER attributes	52
Table 35 – TYPE attributes	53
Table 36 – CHART attributes	53
Table 37 – CYCLE_TIME attribute	54
Table 38 – HEIGHT/WIDTH attribute	54
Table 39 – LENGTH attribute	55
Table 40 – TYPE attribute	55
Table 41 – COLLECTION attributes	56
Table 42 – item-type	56
Table 43 – COMMAND attributes	58
Table 44 – OPERATION attribute	58
Table 45 – TRANSACTION attributes	59
Table 46 – REPLY and REQUEST attributes	60
Table 47 – INDEX attribute	61
Table 48 – BLOCK_B attribute	61
Table 49 – NUMBER attribute	61
Table 50 – SLOT attribute	62
Table 51 – CONNECTION attribute	62
Table 52 – HEADER attribute	62
Table 53 – MODULE attribute	63
Table 54 – CONNECTION attribute	63
Table 55 – APPINSTANCE attribute	63
Table 56 – DOMAIN attributes	64
Table 57 – HANDLING attribute	64
Table 58 – EDIT_DISPLAY attributes	65
Table 59 – EDIT_ITEMS attribute	65
Table 60 – DISPLAY_ITEM attributes	66
Table 61 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS attribute	67
Table 62 – FILE attributes	67
Table 63 – GRAPH attributes	68
Table 64 – HEIGHT/WIDTH attribute	68
Table 65 – CYCLE_TIME attribute	69
Table 66 – X_AXIS attribute	69
Table 67 – GRID attributes	70
Table 68 – VECTORS attribute	70
Table 69 – HANDLING attribute	71
Table 70 – HEIGHT/WIDTH attribute	71
Table 71 – ORIENTATION attribute	72
Table 72 – VALIDITY attributes	72
Table 73 – IMAGE attributes	72
Table 74 – PATH attribute	73
Table 75 – LINK attribute	73
Table 76 – VALIDITY attributes	73

Table 77 – Importing Device Description.....	75
Table 78 – Redefinition attributes	76
Table 79 – Redefinition rules for AXIS attributes.....	76
Table 80 – Redefinition rules for BLOCK_A attributes	77
Table 81 – Redefinition rules for BLOCK_B attributes	77
Table 82 – Redefinition rules for CHART attributes	78
Table 83 – Redefinition rules for COLLECTION attributes	78
Table 84 – Redefinition rules for COMMAND attributes.....	78
Table 85 – Redefinition rules for CONNECTION attributes.....	79
Table 86 – Redefinition rules for DOMAIN attributes	79
Table 87 – Redefinition rules for EDIT_DISPLAY attributes.....	79
Table 88 – Redefinition rules for FILE attributes	79
Table 89 – Redefinition rules for GRAPH attributes.....	80
Table 90 – Redefinition rules for GRID attributes	80
Table 91 – Redefinition rules for IMAGE attributes.....	80
Table 92 – Redefinition rules for LIST attributes	81
Table 93 – Redefinition rules for MENU attributes.....	81
Table 94 – Redefinition rules for METHOD attributes.....	82
Table 95 – Redefinition rules for PROGRAM attributes.....	82
Table 96 – Redefinition rules for RECORD attributes.....	82
Table 97 – Redefinition rules for REFERENCE_ARRAY attributes	82
Table 98 – Redefinition rules for RESPONSE_CODES attributes	83
Table 99 – Redefinition rules for SOURCE attributes	83
Table 100 – Redefinition rules for VALUE_ARRAY attributes	83
Table 101 – Redefinition rules for VARIABLE attributes	84
Table 102 – Redefinition rules for VARIABLE_LIST attributes	85
Table 103 – Redefinition rules for WAVEFORM attributes.....	85
Table 104 – LIKE attributes.....	86
Table 105 – LIST attributes	86
Table 106 – TYPE attribute	86
Table 107 – CAPACITY, COUNT attribute	87
Table 108 – MENU attribute	88
Table 109 – ITEMS attribute.....	89
Table 110 – ACCESS attribute	89
Table 111 – ENTRY attribute.....	90
Table 112 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS attributes.....	90
Table 113 – PURPOSE attribute.....	92
Table 114 – ROLE attribute	93
Table 115 – STYLE attribute	93
Table 116 – VALIDITY attributes	94
Table 117 – METHOD attributes.....	98
Table 118 – ACCESS attributes.....	98
Table 119 – TYPE attributes.....	99

Table 120 – VALIDITY attributes	99
Table 121 – PROGRAM attributes	100
Table 122 – ARGUMENT attribute	100
Table 123 – RECORD attributes	101
Table 124 – REFERENCE_ARRAY attribute	101
Table 125 – ELEMENTS attribute	102
Table 126 – REFRESH attributes	102
Table 127 – UNIT attributes	103
Table 128 – WRITE_AS_ONE attribute	103
Table 129 – RESPONSE_CODES attributes	104
Table 130 – SOURCE attributes	104
Table 131 – EMPHASIS attribute	105
Table 132 – LINE_COLOR attribute	105
Table 133 – LINE_TYPE attribute	106
Table 134 – Y_AXIS attribute	106
Table 135 – VALUE_ARRAY attributes	107
Table 136 – NUMBER_OF_ELEMENTS attribute	108
Table 137 – TYPE attribute	108
Table 138 – VARIABLE attributes	108
Table 139 – CLASS attributes	109
Table 140 – TYPE attributes	110
Table 141 – DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER attributes	112
Table 142 – BIT_ENUMERATED attributes	114
Table 143 – status-class attributes	115
Table 144 – ALL, AO, DV, TV attributes	116
Table 145 – Enumerated types attributes	116
Table 146 – Index type attributes	117
Table 147 – Object reference type attribute	117
Table 148 – DEFAULT_REFERENCE attributes	117
Table 149 – String types attributes	119
Table 150 – CONSTANT_UNIT attribute	119
Table 151 – DEFAULT_VALUE attribute	119
Table 152 – HANDLING attribute	120
Table 153 – INITIAL_VALUE attribute	120
Table 154 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS attributes	121
Table 155 – READ/WRITE_TIMEOUT attributes	123
Table 156 – STYLE attribute	123
Table 157 – VALIDITY attributes	123
Table 158 – VARIABLE_LIST attributes	124
Table 159 – WAVEFORM attributes	124
Table 160 – TYPE attribute	125
Table 161 – XY attribute	125

Table 162 – YT attribute	126
Table 163 – HORIZONTAL attribute	127
Table 164 – VERTICAL attribute	127
Table 165 – EMPHASIS attribute	127
Table 166 – HANDLING attribute	128
Table 167 – EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS attribute	128
Table 168 – KEY_POINTS attribute	129
Table 169 – X_VALUES, Y_VALUES attribute	129
Table 170 – LINE_COLOR attribute	130
Table 171 – LINE_TYPE attribute	130
Table 172 – Y_AXIS attribute	131
Table 173 – DEFINITION attributes	131
Table 174 – HELP attribute	132
Table 175 – LABEL attribute	132
Table 176 – MEMBERS attributes	133
Table 177 – RESPONSE_CODES attribute	133
Table 178 – OPEN and CLOSE attributes	134
Table 179 – IF, SELECT conditional	135
Table 180 – Referencing an EDD instance	136
Table 181 – Referencing elements of VARIABLE	136
Table 182 – Referencing elements of RECORD	136
Table 183 – Referencing elements of VALUE_ARRAY	137
Table 184 – Referencing members of COLLECTION	137
Table 185 – Referencing members of REFERENCE_ARRAY	137
Table 186 – Referencing members of VARIABLE_LISTS	138
Table 187 – Referencing members of a BLOCK_A PARAMETERS	138
Table 188 – Referencing members of BLOCK_A PARAMETER_LISTS	138
Table 189 – Referencing members of BLOCK_A LOCAL_PARAMETER	138
Table 190 – Referencing BLOCK_A CHARACTERISTICS	139
Table 191 – Referencing members of FILE	139
Table 192 – Referencing elements of LIST	139
Table 193 – Referencing members of CHART	140
Table 194 – Referencing members of GRAPH	140
Table 195 – Referencing members of SOURCE	140
Table 196 – Referencing AXIS of a GRAPH. SOURCE, WAVEFORM	141
Table 197 – String as a string literal	141
Table 198 – String as a string variable	141
Table 199 – String as an enumeration value	142
Table 200 – String as a dictionary reference	142
Table 201 – Referencing HELP and LABEL attributes of EDD instances	142
Table 202 – String operation	143
Table 203 – Format specifier	143
Table 204 – Primary expressions	144

Table 205 – Attribute values of VARIABLES	144
Table 206 – AXIS Attribute Values	145
Table 207 – LIST Attribute Values	145
Table 208 – Unary expressions	145
Table 209 – Multiplicative operators	146
Table 210 – Additive operators	146
Table 211 – Shift operators	147
Table 212 – Relational operators	147
Table 213 – Equality operators	147
Table 214 – Text dictionary attributes	149
Table A.1 – Conventions for integer constants	154
Table A.2 – Using escape sequences in string literals	155
Table A.3 – Using language codes in string literals	155
Table A.4 – EDDL operators	156
Table A.5 – EDDL keywords	156
Table B.1 – Format for the Builtins lexical element tables	224
Table B.2 – Contents of the lexical element table	224
Table B.3 – Builtin abort	225
Table B.4 – Builtin abort_on_all_comm_errors	225
Table B.5 – Builtin ABORT_ON_ALL_COMM_STATUS	226
Table B.6 – Builtin ABORT_ON_ALL_DEVICE_STATUS	226
Table B.7 – Builtin ABORT_ON_ALL_RESPONSE_CODES	227
Table B.8 – Builtin abort_on_all_response_codes	227
Table B.9 – Builtin abort_on_comm_error	227
Table B.10 – Builtin ABORT_ON_COMM_ERROR	228
Table B.11 – Builtin ABORT_ON_COMM_STATUS	228
Table B.12 – Builtin ABORT_ON_DEVICE_STATUS	229
Table B.13 – Builtin ABORT_ON_NO_DEVICE	229
Table B.14 – Builtin ABORT_ON_RESPONSE_CODE	230
Table B.15 – Builtin abort_on_response_code	231
Table B.16 – Builtin abs	231
Table B.17 – Builtin ACKNOWLEDGE	232
Table B.18 – Builtin acknowledge	232
Table B.19 – Builtin acos	232
Table B.20 – Builtin add_abort_method	233
Table B.21 – Builtin add_abort_method	233
Table B.22 – Builtin asin	234
Table B.23 – Builtin assign	234
Table B.24 – Builtin assign_double	235
Table B.25 – Builtin assign_float	235
Table B.26 – Builtin assign_int	235
Table B.27 – Builtin assign_var	236
Table B.28 – Builtin atan	236

Table B.29 – Builtin ATOF	236
Table B.30 – Builtin atof	237
Table B.31 – Builtin ATOI	237
Table B.32 – Builtin atoi	237
Table B.33 – Builtin BUILD_MESSAGE	238
Table B.34 – Builtin cbrt	238
Table B.35 – Builtin ceil	238
Table B.36 – Builtin cos	239
Table B.37 – Builtin cosh	239
Table B.38 – Builtin dassign	239
Table B.39 – Builtin Date_to_DayOfMonth	240
Table B.40 – Builtin Date_to_Month	240
Table B.41 – Builtin Date_to_Year	240
Table B.42 – Builtin DELAY	241
Table B.43 – Builtin delay	241
Table B.44 – Builtin DELAY_TIME	242
Table B.45 – Builtin delayfor	242
Table B.46 – Builtin DICT_ID	243
Table B.47 – Builtin discard_on_exit	243
Table B.48 – Builtin display	244
Table B.49 – Builtin display_builtin_error	244
Table B.50 – Builtin display_comm_error	245
Table B.51 – Builtin display_comm_status	245
Table B.52 – Builtin display_device_status	245
Table B.53 – Builtin display_dynamics	246
Table B.54 – Builtin display_message	246
Table B.55 – Builtin display_response_code	247
Table B.56 – Builtin display_response_status	248
Table B.57 – Builtin display_xmtr_status	248
Table B.58 – Builtin edit_device_value	249
Table B.59 – Builtin edit_local_value	250
Table B.60 – Builtin exp	250
Table B.61 – Builtin ext_send_command	251
Table B.62 – Builtin ext_send_command_trans	251
Table B.63 – Builtin fail_on_all_comm_errors	252
Table B.64 – Builtin fail_on_all_response_codes	252
Table B.65 – Builtin fail_on_comm_error	253
Table B.66 – Builtin fail_on_response_code	253
Table B.67 – Builtin fassign	254
Table B.68 – Builtin fgetval	254
Table B.69 – Builtin float_value	254
Table B.70 – Builtin floor	255
Table B.71 – Builtin fmod	255

Table B.72 – Builtin fsetval	256
Table B.73 – Builtin ftoa	256
Table B.74 – Builtin fvar_value	256
Table B.75 – Builtin get_acknowledgement	257
Table B.76 – Builtin get_comm_error	258
Table B.77 – Builtin get_comm_error_string	258
Table B.78 – Builtin get_date	259
Table B.79 – Builtin get_date_value	259
Table B.80 – Builtin get_dds_error	260
Table B.81 – Builtin GET_DEV_VAR_VALUE	260
Table B.82 – Builtin get_dev_var_value	261
Table B.83 – Builtin get_dictionary_string	262
Table B.84 – Builtin get_double	262
Table B.85 – Builtin get_double_value	263
Table B.86 – Builtin get_enum_string	263
Table B.87 – Builtin get_float	264
Table B.88 – Builtin get_float_value	264
Table B.89 – Builtin GET_LOCAL_VAR_VALUE	265
Table B.90 – Builtin get_local_var_value	265
Table B.91 – Builtin get_more_status	266
Table B.92 – Builtin get_resolve_status	266
Table B.93 – Builtin get_response_code	267
Table B.94 – Builtin get_response_code_string	268
Table B.95 – Builtin get_signed	268
Table B.96 – Builtin get_signed_value	269
Table B.97 – Builtin get_status_code_string	269
Table B.98 – Builtin get_status_string	270
Table B.99 – Builtin get_stddict_string	270
Table B.100 – Builtin get_string	271
Table B.101 – Builtin get_string_value	271
Table B.102 – Builtin GET_TICK_COUNT	272
Table B.103 – Builtin get_unsigned	272
Table B.104 – Builtin get_unsigned_value	273
Table B.105 – Builtin iassign	273
Table B.106 – Builtin igetval	274
Table B.107 – Builtin IGNORE_ALL_COMM_STATUS	274
Table B.108 – Builtin IGNORE_ALL_DEVICE_STATUS	275
Table B.109 – Builtin IGNORE_ALL_RESPONSE_CODES	276
Table B.110 – Builtin IGNORE_COMM_ERROR	276
Table B.111 – Builtin IGNORE_COMM_STATUS	277
Table B.112 – Builtin IGNORE_DEVICE_STATUS	277
Table B.113 – Builtin IGNORE_NO_DEVICE	278
Table B.114 – Builtin IGNORE_RESPONSE_CODE	278

Table B.115 – Builtin int_value	278
Table B.116 – Builtin is_NaN	279
Table B.117 – Builtin isetval	279
Table B.118 – Builtin ITEM_ID	280
Table B.119 – Builtin ITOA	280
Table B.120 – Builtin itoa	280
Table B.121 – Builtin ivar_value	281
Table B.122 – Builtin lassign	281
Table B.123 – Builtin lgetval	281
Table B.124 – Builtin ListDeleteElementAt	282
Table B.125 – Builtin ListInsert	282
Table B.126 – Builtin log	283
Table B.127 – Builtin log10	283
Table B.128 – Builtin log2	284
Table B.129 – Lexical elements of Builtin LOG_MESSAGE	284
Table B.130 – Builtin long_value	284
Table B.131 – Builtin lsetval	285
Table B.132 – Builtin lvar_value	285
Table B.133 – Builtin MEMBER_ID	286
Table B.134 – Builtin MenuDisplay	286
Table B.135 – Builtin method_abort	287
Table B.136 – Builtin process_abort	287
Table B.137 – Builtin pow	287
Table B.138 – Builtin process_abort	288
Table B.139 – Builtin put_date	288
Table B.140 – Builtin put_date_value	289
Table B.141 – Builtin put_double	289
Table B.142 – Builtin put_double_value	290
Table B.143 – Builtin put_float	290
Table B.144 – Builtin put_float_value	291
Table B.145 – Builtin PUT_MESSAGE	291
Table B.146 – Builtin put_message	292
Table B.147 – Builtin put_signed	292
Table B.148 – Builtin put_signed_value	293
Table B.149 – Builtin put_string	294
Table B.150 – Builtin put_string_value	294
Table B.151 – Builtin put_unsigned	295
Table B.152 – Builtin put_unsigned_value	296
Table B.153 – Lexical elements of Builtin READ_COMMAND	296
Table B.154 – Builtin read_value	297
Table B.155 – Builtin remove_abort_method	297
Table B.156 – Builtin remove_abort_method	298
Table B.157 – Builtin remove_all_abort_methods	298

Table B.158 – Builtin resolve_array_ref	299
Table B.159 – Builtin resolve_block_ref	299
Table B.160 – Builtin resolve_param_list_ref	300
Table B.161 – Builtin resolve_parm_ref	300
Table B.162 – Builtin resolve_record_ref	301
Table B.163 – Builtin retry_on_all_comm_errors	301
Table B.164 – Builtin RETRY_ON_ALL_COMM_STATUS	302
Table B.165 – Builtin RETRY_ON_ALL_DEVICE_STATUS	302
Table B.166 – Builtin RETRY_ON_ALL_RESPONSE_CODES	303
Table B.167 – Builtin retry_on_all_response_codes	303
Table B.168 – Builtin RETRY_ON_COMM_ERROR	304
Table B.169 – Builtin retry_on_comm_error	304
Table B.170 – Builtin RETRY_ON_COMM_STATUS	305
Table B.171 – Builtin RETRY_ON_DEVICE_STATUS	305
Table B.172 – Builtin RETRY_ON_NO_DEVICE	306
Table B.173 – Builtin RETRY_ON_RESPONSE_CODE	307
Table B.174 – Builtin retry_on_response_code	307
Table B.175 – Builtin round	307
Table B.176 – Builtin rspcode_string	308
Table B.177 – Builtin save_on_exit	309
Table B.178 – Builtin save_values	309
Table B.179 – Builtin SELECT_FROM_LIST	309
Table B.180 – Builtin select_from_list	310
Table B.181 – Builtin select_from_menu	311
Table B.182 – Builtin send	311
Table B.183 – Builtin send_all_values	312
Table B.184 – Builtin send_command	312
Table B.185 – Builtin send_command_trans	313
Table B.186 – Builtin send_on_exit	314
Table B.187 – Builtin send_trans	314
Table B.188 – Builtin send_value	315
Table B.189 – Builtin SET_NUMBER_OF_RETRIES	315
Table B.190 – Builtin sin	315
Table B.191 – Builtin sinh	316
Table B.192 – Builtin sqrt	316
Table B.193 – Builtin strcmp	316
Table B.194 – Builtin strlen	317
Table B.195 – Builtin strlenr	317
Table B.196 – Builtin strmid	318
Table B.197 – Builtin strstr	318
Table B.198 – Builtin strtrim	318
Table B.199 – Builtin strupr	319
Table B.200 – Builtin tan	319

Table B.201 – Builtin tanh	319
Table B.202 – Builtin To_Date_and_Time	320
Table B.203 – Builtin trunc	320
Table B.204 – Builtin VARID	321
Table B.205 – Builtin vassign	321
Table B.206 – Builtin WRITE_COMMAND	321
Table B.207 – Builtin XMTR_ABORT_ON_ALL_COMM_STATUS	322
Table B.208 – Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS	323
Table B.209 – Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES	323
Table B.210 – Builtin XMTR_ABORT_ON_COMM_ERROR	324
Table B.211 – Builtin XMTR_ABORT_ON_COMM_STATUS	324
Table B.212 – Builtin XMTR_ABORT_ON_DATA	325
Table B.213 – Builtin XMTR_ABORT_ON_DEVICE_STATUS	325
Table B.214 – Builtin XMTR_ABORT_ON_NO_DEVICE	326
Table B.215 – Builtin XMTR_ABORT_ON_RESPONSE_CODE	326
Table B.216 – Builtin XMTR_IGNORE_ALL_COMM_STATUS	327
Table B.217 – Builtin XMTR_IGNORE_ALL_DEVICE_STATUS	327
Table B.218 – Builtin XMTR_IGNORE_ALL_RESPONSE_CODES	328
Table B.219 – Builtin XMTR_IGNORE_COMM_ERROR	328
Table B.220 – Builtin XMTR_IGNORE_COMM_STATUS	329
Table B.221 – Builtin XMTR_IGNORE_DEVICE_STATUS	329
Table B.222 – Builtin XMTR_IGNORE_NO_DEVICE	330
Table B.223 – Builtin XMTR_IGNORE_RESPONSE_CODE	330
Table B.224 – Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS	331
Table B.225 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE	332
Table B.226 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES	332
Table B.227 – Builtin XMTR_RETRY_ON_COMM_ERROR	333
Table B.228 – Builtin XMTR_RETRY_ON_COMM_STATUS	333
Table B.229 – Builtin XMTR_RETRY_ON_DATA	334
Table B.230 – Builtin XMTR_RETRY_ON_DEVICE_STATUS	334
Table B.231 – Builtin XMTR_RETRY_ON_NO_DEVICE	335
Table B.232 – Builtin XMTR_RETRY_ON_RESPONSE_CODE	335
Table B.233 – Builtin YearMonthDay_to_Date	336
Table B.234 – Contents of the return codes description table	336
Table B.235 – Return Code Description	336
Table D.1 – Profile selection tables	350
Table D.2 – EDDL Formal Definition profile tables	350
Table D.3 – Contents of selection tables	350
Table D.4 – EDDL element selection for PROFIBUS	351
Table D.5 – Builtin profile for PROFIBUS	354
Table D.6 – EDDL element selection for Fieldbus Foundation	357
Table D.7 – Builtin profile for Fieldbus Foundation	361
Table D.8 – EDDL element selection for HCF	365

Table D.9 – Builtin profile for HCF	368
Table D.10 – METHOD DEFINITIONS data types	372
Table D.11 – VARIABLE TYPES.....	372
Table D.12 – DATE coding	373
Table D.13 – DATE_AND_TIME coding	373
Table D.14 – DURATION coding	374
Table D.15 – TIME coding	374
Table D.16 – TIME_VALUE coding	374
Table D.17 – PACKED_ASCII coding.....	375

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2006

Withd2Wm

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FUNCTION BLOCKS (FB) FOR PROCESS CONTROL –

Part 3: Electronic Device Description Language (EDDL)

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
- 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.

IEC takes no position concerning the evidence, validity and scope of this patent right. The holder of this patent right has assured the IEC that he is willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with IEC.

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

This International Standard has been prepared by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

This first edition cancels and replaces the EDDL specification given in the first edition of IEC 61804-2, published in 2004¹.

This bilingual version (2012-12) corresponds to the monolingual English version, published in 2006-09.

¹ The historical background to the EDDL specification is given in Annex E.

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

CDV	Report on voting
65C/406/CDV	65C/421/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.

The French version of this standard has not been voted upon.

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

The list of all the parts of the IEC 61804 series, under the general title *Function Blocks (FB) for process control*, can be found on the IEC website.

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

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

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

INTRODUCTION

The EDDL fills the gap between the conceptual FB 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.

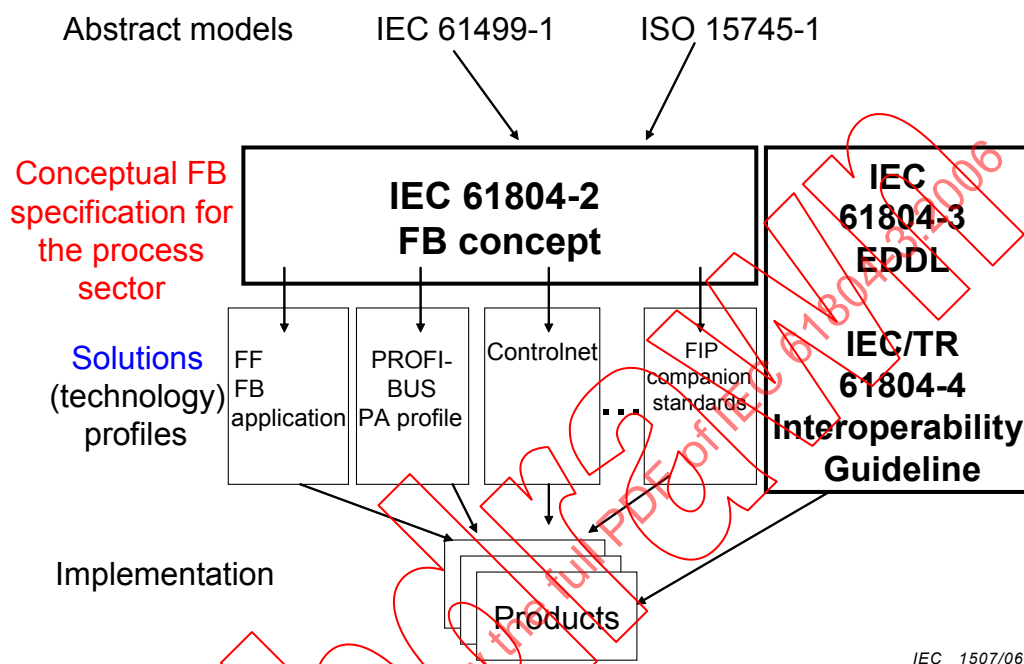


Figure 1 – Position of the IEC 61804 series related to other standards and products

FUNCTION BLOCKS (FB) FOR PROCESS CONTROL –

Part 3: Electronic Device Description Language (EDDL)

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 standard 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 to be used to create Electronic Device Description (EDD). This EDD is used with appropriate tools to generate interpretative code to support parameter handling, operation, and monitoring of automation system components such as remote I/Os, controllers, sensors, and programmable controllers. Tool implementation is outside the scope of this standard.

This standard 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.

NOTE 1 The EDDL may also be used for the description of product properties in other domains.

The EDDL and the device-related EDD is applicable to industrial automation.

NOTE 2 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.

This International Standard satisfies the requirements of Clause 9 of IEC 61804-1.

2 Normative references

The following referenced documents are indispensable for the application 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 61499-1:2005, *Function blocks – Part 1: Architecture*

IEC 61804-1:2003, *Function blocks (FB) for process control – Part 1: Overview of system aspects*

ISO/IEC 2022, *Information technology – Character code structure and extension techniques*

ISO/IEC 2375:2003, *Information technology – Procedure for registration of escape sequences and coded character sets*

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

ISO/IEC 8859-1:1998, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 9899, *Programming languages – C*

ISO/IEC 10646-1:2000, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

ISO 639, *Codes for the representation of names of languages*

ISO 3166, *Codes for the representation of names of countries and their subdivisions*

IEEE 754:1985 (R1990), *Binary Floating-Point Arithmetic*

3 Terms, definitions, abbreviated terms and acronyms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions, some of which have been compiled from the referenced documents, apply.

NOTE The terms in IEC 60050-351 apply partially.

3.2

algorithm

finite set of well-defined rules for the solution of a problem in a finite number of operations

3.3

attribute

description of an externally visible characteristic or feature of an object

NOTE The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behaviour of an object. Attributes are divided into class attributes and instance attributes.

3.4

Builtin

predefined subroutine for communication and display executed by the EDD application

3.5

configuration (of a system or device)

step in system design: selecting functional units, assigning their locations and defining their interconnections

[IEC 61499-1]

3.6**data**

representation of facts, concepts or instructions in a formalized manner suitable for communication, interpretation or processing by human beings or by automatic means

[ISO/AFNOR Dictionary of Computer Science]

3.7**data connection**

association established between functional units for conveyance of data

[IEC 61499-1]

3.8**data input**

interface of an FB which receives data from a data connection

[IEC 61499-1]

3.9**data output**

interface of an FB, which supplies data to a data connection

[IEC 61499-1]

3.10**data type**

set of values together with a set of permitted operations

[ISO 2382 series]

3.11**device**

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

[IEC 61499-1]

3.12**EDD application**

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

3.13**EDDL processor**

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

3.14**EDDL profile**

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

3.15**Electronic Device Description Language****EDDL**

methodology for describing parameter(s) of an automation system component

3.16

Electronic Device Description

EDD

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

NOTE The Electronic Device Description is created using the Electronic Device Description Language (EDDL)

3.17

Electronic Device Description Source

EDDS

ASCII file containing a specific device description

3.18

Electronic Device Description Technology

EDDT

technology which includes the EDD development process, the EDD usage and the involved tool chain

3.19

Electronic Device Description Checker

test tool which checks the Lexical Structure and partly the semantic of EDD sources to guarantee compliance of EDD sources with the EDD language

3.20

Electronic Device Description Compiler

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

3.21

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.

3.22

entity

particular thing, such as a person, place, process, object, concept, association, or event

[IEC 61499-1]

3.23

event

instantaneous occurrence that is significant to scheduling the execution of an algorithm

[IEC 61499-1]

NOTE The execution of an algorithm may make use of variables associated with an event.

3.24

exception

event that causes suspension of normal execution

[IEC 61499-1]

3.25**function**

specific purpose of an entity or its characteristic action

[IEC 61499-1]

3.26**functional unit**

entity of hardware or software, or both, capable of accomplishing a specified purpose

[ISO/AFNOR Dictionary of Computer Science]

3.27**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

[IEC 61499-1]

NOTE Typical operations of a FB include modification of the values of the data in its associated data structure.

3.28**hardware**

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

[ISO/AFNOR Dictionary of Computer Science]

3.29**implementation**

development phase in which the hardware and software of a system become operational

[IEC 61499-1]

3.30**input variable**

variable whose value is supplied by a data input, and which may be used in one or more operations of a FB

NOTE An input parameter of an FB, as defined in IEC 61131-3, is an input variable.

[IEC 61499-1]

3.31**instance**

functional unit comprising an individual, named entity with the attributes of a defined type

[IEC 61499-1]

3.32**instance name**

identifier associated with, and designating, an instance

[IEC 61499-1]

3.33**instantiation**

creation of an instance of a specified type

[IEC 61499-1]

3.34

interface

shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

[IEC 60050-351:1998, 11-19]

3.35

internal variable

variable whose value is used or modified by one or more operations of an FB but is not supplied by a data input or to a data output

[IEC 61499-1]

3.36

invocation

process of initiating the execution of the sequence of operations specified in an algorithm

[IEC 61499-1]

3.37

mapping

set of values having defined correspondence with the quantities or values of another set

[ISO/AFNOR Dictionary of Computer Science]

3.38

model

representation of a real world process, device, or concept

[IEC 61499-1]

3.39

operation

well-defined action that, when applied to any permissible combination of known entities, produces a new entity

[ISO/AFNOR Dictionary of Computer Science]

3.40

output variable

variable whose value is established by one or more operations of a FB, and is supplied to a data output

NOTE An output parameter of an FB, as defined in IEC 61131-3, is an output variable.

[IEC 61499-1]

3.41

parameter

variable that is given a constant value for a specified application and that may denote the application

[ISO/AFNOR Dictionary of Computer Science]

3.42

preprocessor

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

3.43 **preprocessor directives**

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

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

3.44 **resource**

functional unit contained within a device which has independent control of its operation, and which provides various services to applications, including the scheduling and execution of algorithms

NOTE 1 The RESOURCE defined in IEC 61131-3 is a programming language element corresponding to the resource defined above.

NOTE 2 A device contains one or more resources.

3.45 **resource management application**

application whose primary function is the management of a single resource

[IEC 61499-1]

3.46 **service**

functional capability of a resource, which can be modelled by a sequence of service primitives

[IEC 61499-1]

3.47 **software**

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

[IEC 61499-1]

3.48 **system**

set of interrelated elements considered in a defined context as a whole and separated from its environment

[IEV 350-11-01:1998]

NOTE 1 Such elements may be both material objects and concepts as well as the results thereof (for example, forms of organization, mathematical methods, and programming languages).

NOTE 2 The system is considered to be separated from the environment and other external systems by an imaginary surface, which can cut the links between them and the considered system.

3.49 **text dictionary**

collection of multilingual or other texts within the EDD

NOTE References within an EDD are used to select an appropriate text dictionary.

3.50 **type**

software element, which specifies the common attributes shared by all instances of the type

[IEC 61499-1]

3.51

type name

identifier associated with, and designating, a type

[IEC 61499-1]

3.52

variable

software entity that may take different values, one at a time

NOTE 1 The values of a variable are usually restricted to a certain data type.

NOTE 2 Variables are described as input variables, output variables, and internal variables.

[IEC 61499-1]

3.53 Abbreviated terms and acronyms

ANSI	American National Standard Institute
ANSI C	American National Standard Institute for the programming language C (see ISO/IEC 9899)
ASCII	American Standard Code for Information Interchange (see ISO/IEC 10646-1)
ASN.1	Abstract Lexical Structure Notation 1
DD	Device Description
EDD	Electronic Device Description
EDDI	Electronic Device Description Interpreter
EDDL	Electronic Device Description Language
EUC	Extended Unit Code (see ISO/IEC 2022)
FB	Function Block
HMI	Human Machine Interface
HTML	Hypertext Mark-up Language
I/O	Input/Output
ID	Identifier
OSI	Open Systems Interconnection
wao	Write as one

4 Conformance statement

A statement of conformance to a profile of this International Standard shall be stated² as either

– compliance to IEC 61804-3:2006 Annex D.n <Type>

or

– compliance to IEC 61804-3/Ed1.0 Annex D.n <Type>

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

NOTE The Type may be, for example, the name of the communication protocol used like PROFIBUS, Foundation Fieldbus, or HART.

² In accordance with the ISO/IEC Directives.

Product standards shall not include any conformity assessment aspects (including QM provisions), neither normative nor informative, other than provisions for product testing (evaluation and examination).

5 Conventions for lexical structures

The EDDL is generally described using lexical structures in which the elements and the presence of fields are specified. A general form of the lexical structures is shown below.

ABC field1, field2

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

Field1 and field2 are fields of the lexical element ABC. Each field is mandatory and may have more than one attribute. If a field has attributes, the presence of the attributes is specified in a table. A comma separates Field1 and 2. The comma is a lexical element and is not coded explicitly.

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

Table 1 – Field attribute descriptions

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

The characters in the Usage column have the following meanings.

- m: this attribute is mandatory and shall be present.
- o: this attribute is optional and need not be present.
- s: this attribute is a selection. One, and only one, of the fields marked with s (Z1 or Z2) shall be present.
- c: this attribute is conditional. The condition is described in the Description column.

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

ABC field1+

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

ABC field2*

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

ABC [field1, field2]+

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

ABC field1, (field2, field3)<exp>

<exp> indicates that field2 and field3 are used in conjunction with the conditional expression, (conditional constructs are specified in 7.32). The expression <exp> refers only to the fields within the preceding curved brackets (). Usage of conditional expressions is optional.

6 EDD and EDDL model

6.1 Overview of EDD and EDDL

An Electronic Device Description (EDD) contains all device 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.

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

6.2 EDD architecture

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

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

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

6.3 Concepts of EDD

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

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

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

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

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

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

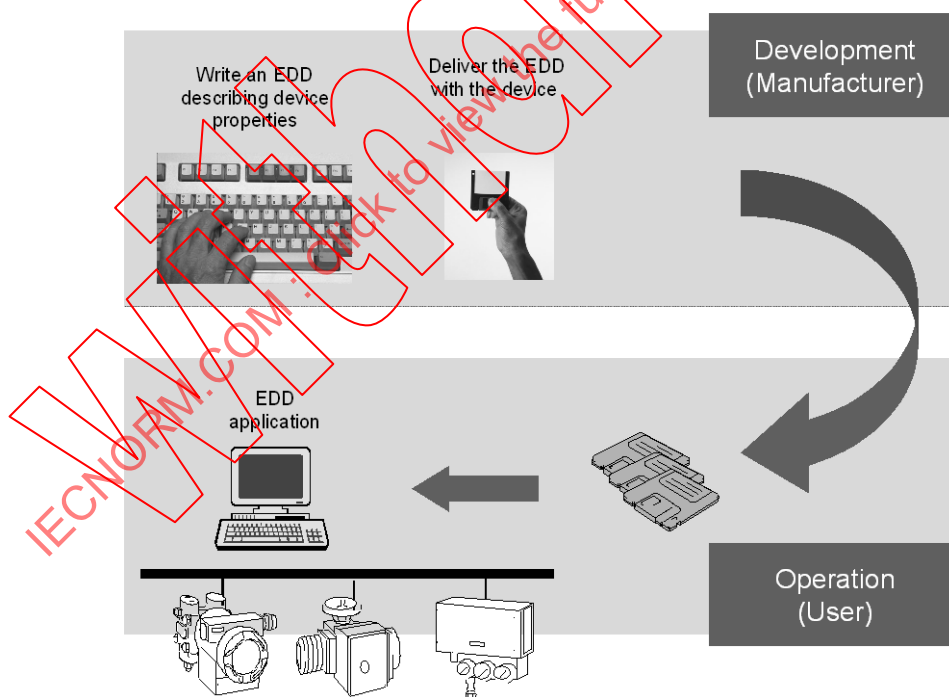
6.4 Principles of the EDD development process

6.4.1 General

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

6.4.2 EDD source generation

The EDD generation process is shown in Figure 2.



IEC 1511/06

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 device memory or delivered using separate storage media or downloaded from an appropriate network server. The EDD is 'interpreted' or 'browsed' by an EDD-application. EDDs are normally stored as source files or preprocessed files.

6.4.3 EDD preprocessing

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

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

6.4.4 EDD compilation

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

6.5 Interrelations between the lexical structure and formal definitions

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

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

6.6 Builtins

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

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

The library of Builtins is defined in Annex B.

6.7 Profiles

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

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

7 Electronic Device Description Language

7.1 Overview

7.1.1 EDDL features

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

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

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

7.1.2 Syntax representation

The specification of the EDDL language in this clause of the standard uses an abstract syntax. The abstract lexical structure in this section 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 the annex).

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

7.1.3 EDD language elements

The language is structured in the following language elements:

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

Frequently used text strings, for example, help text and multi-lingual lists of labels should be separated in a text dictionary (see 7.36).

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

7.1.4 Basic construction elements

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

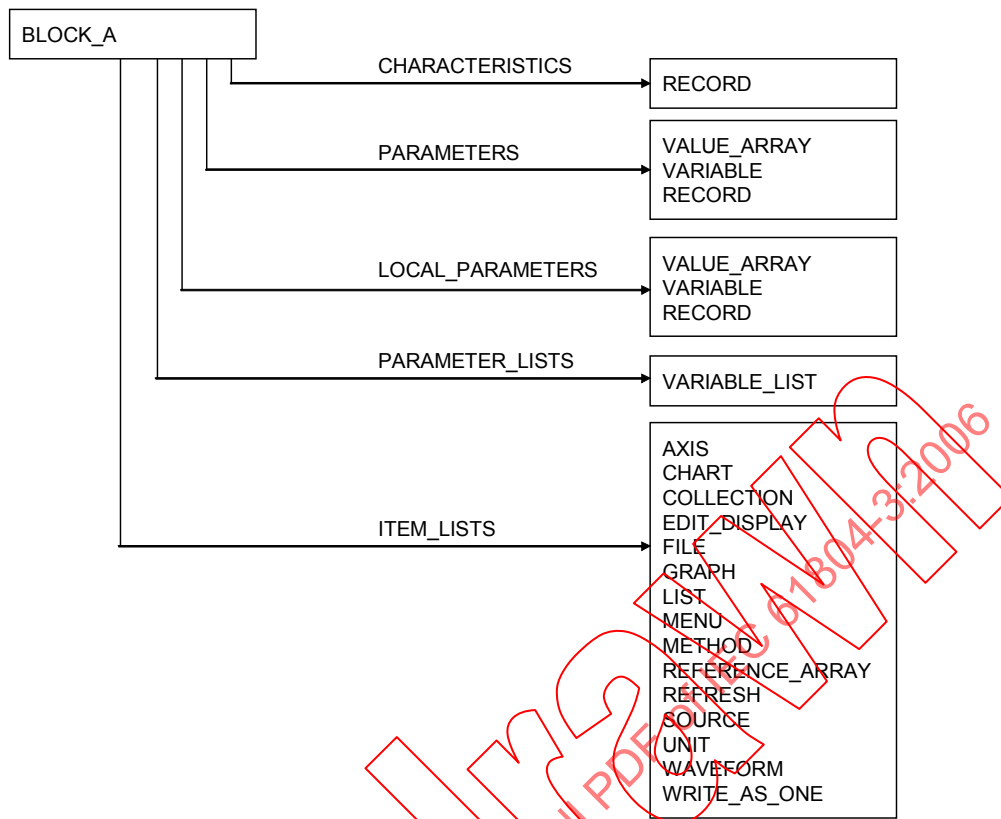
Some constructs have similar names and functions while differing in their detailed specification. This additional variety has been included to ensure compatibility with several existing description languages. Appropriate mapping cross-references are given by profiles.

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.

Some basic constructs refer to other basic constructs. In the following list, the basic constructs as well as the relationships between the basic constructs are shown:

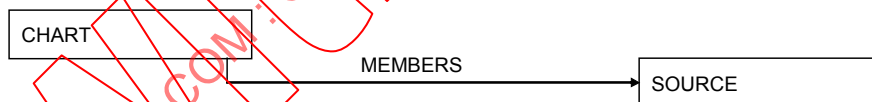
- **AXIS** describes the axis of a CHART or GRAPH (see 7.3).
- **BLOCK_A** is a logical grouping of CHARACTERISTICS, PARAMETERS, PARAMETER_LISTS, and ITEM_LISTS, see Figure 3. To access one item of BLOCK_A, the instance of the block should be used (see 7.4.1 and Figure 3).



IEC 1512/06

Figure 3 – BLOCK_A

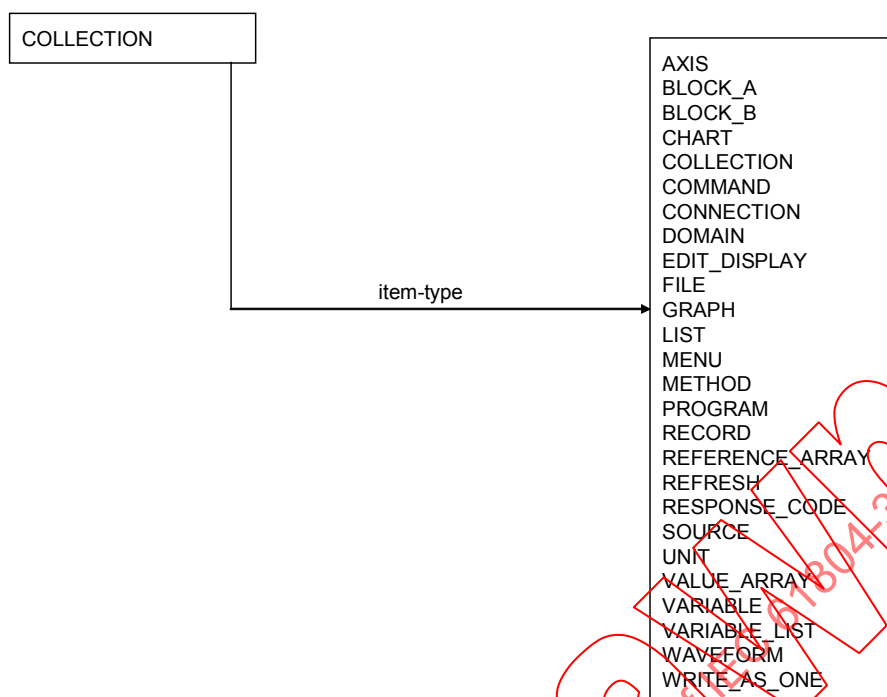
- **BLOCK_B** is used to structure variables in instances of logical block types. For accessing a VARIABLE within a BLOCK_B construct, the basic construct COMMAND is used to provide the relative addressing (see 7.4.2).
- **CHART** describes a chart used to display data from a device in the EDD application (see 7.5 and Figure 4).



IEC 1513/06

Figure 4 – CHART

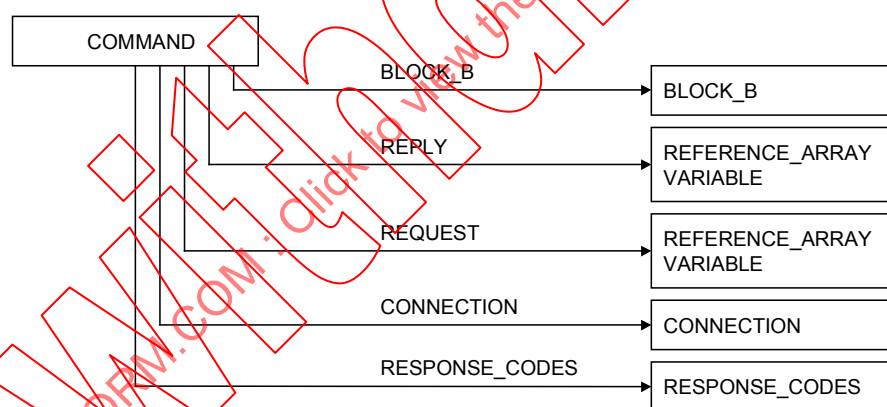
- **COLLECTION** describes logical groupings of data (see 7.6 and Figure 5).



IEC 1514/06

Figure 5 – COLLECTION

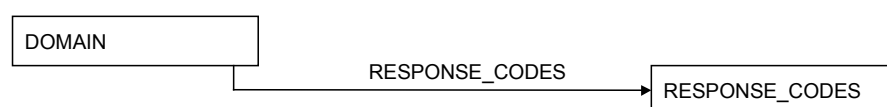
- **COMMAND** describes the structure and the addressing of the variables in the device (see 7.7 and Figure 6).



IEC 1515/06

Figure 6 – COMMAND

- **CONNECTION** defines multiple applications in a device (see 7.8).
- **DOMAIN** can be used to download or upload moderately large amounts of data to or from a device (see 7.9 and Figure 7).



IEC 1516/06

Figure 7 – DOMAIN

- **EDIT_DISPLAY** describes how the data will be represented to a user by a display device (see 7.10 and Figure 8).

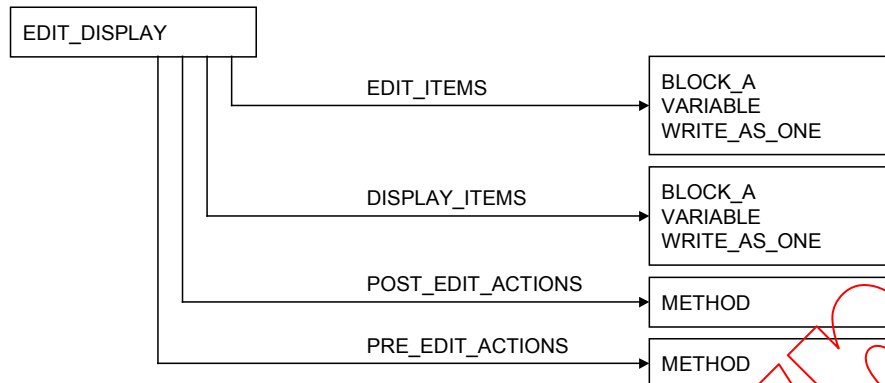


Figure 8 – EDIT_DISPLAY

IEC 1517/06

- **FILE** describes an area of persistent storage (see 7.11 and Figure 9).

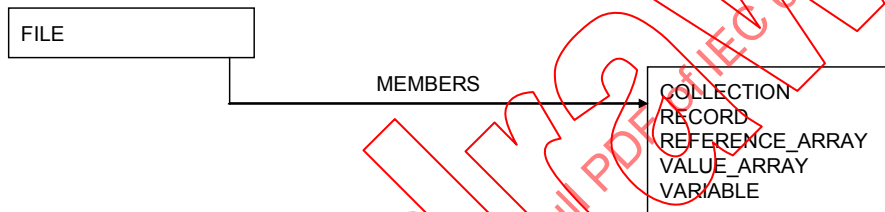


Figure 9 – FILE

IEC 1518/06

- **GRAPH** describes a graph used to display data from a device in the EDD application (see 7.12 and Figure 10).

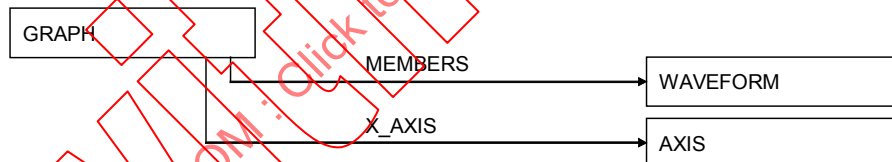
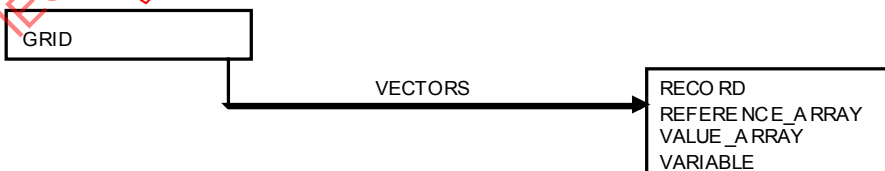


Figure 10 – GRAPH

IEC 1519/06

- **GRID** describes a matrix used to display data from a device in the EDD application (see 7.13 and Figure 11).



IEC 1520/06

Figure 11 – GRID

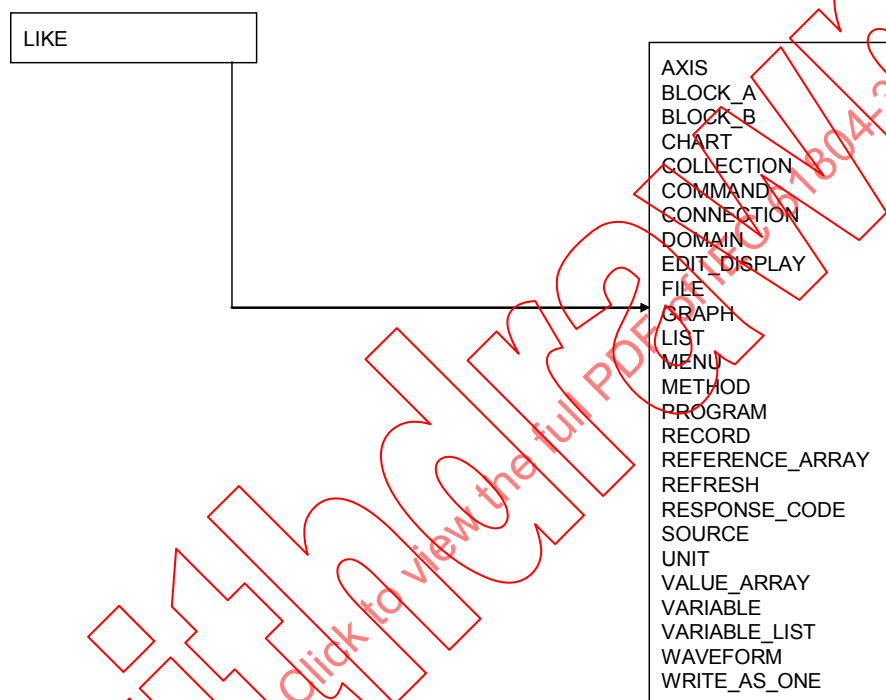
- **IMAGE** describes a visual image (see 7.13 and Figure 12).



IEC 1521/06

Figure 12 – IMAGE

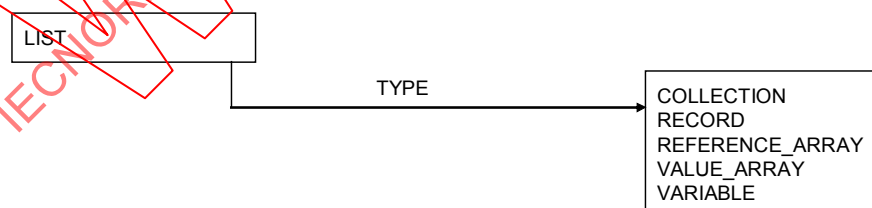
- **IMPORT** is used to import an EDD and to modify its existing definitions (see 7.13).
- **LIKE** creates a new instance of an existing instance of a basic construct. The attributes of the new instance may be redefined, added or deleted (see 7.16 and Figure 13).



IEC 1522/06

Figure 13 – LIKE

- **LIST** describes a sequence of elements (see 7.17 and Figure 14).



IEC 1523/06

Figure 14 – LIST

- **MENU** describes how the data will be presented to a user by an EDD application (see 7.18 and Figure 15).

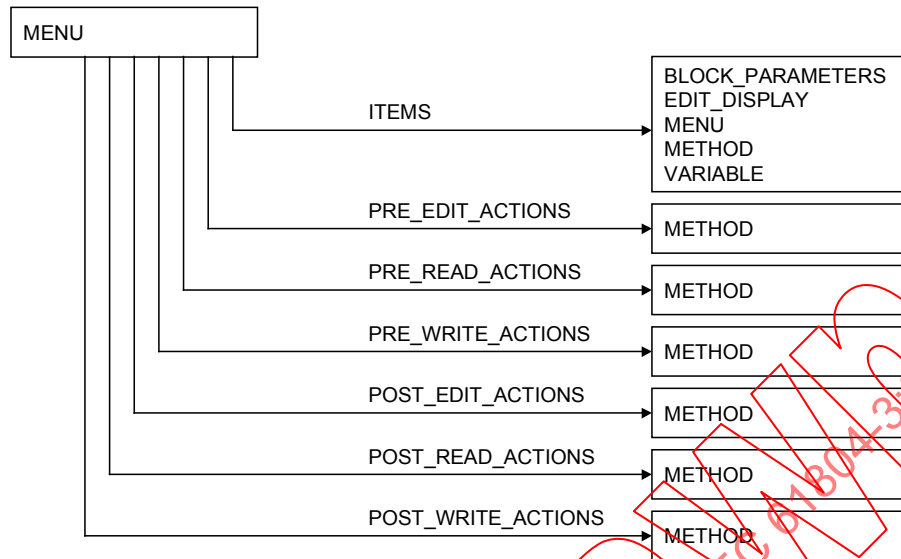


Figure 15 – MENU

IEC 1524/06

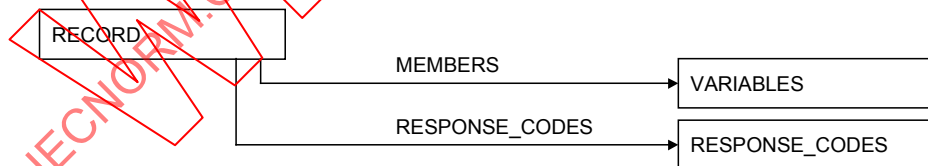
- **METHOD** describes the execution of complex sequence of event interactions that shall take place between system devices such as display units, configurators and field devices (see 7.19).
- **PROGRAM** specifies how device executable code can be initiated by an appropriate agent (see 7.20 and Figure 16).



IEC 1525/06

Figure 16 – PROGRAM

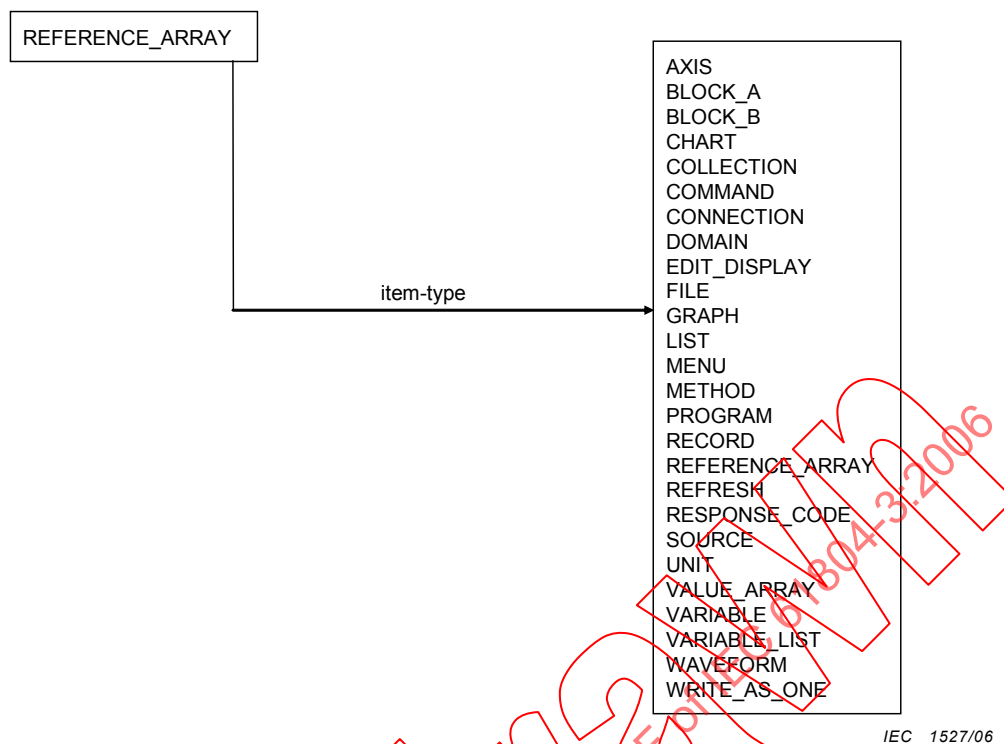
- **RECORD** describes the data contained in the device (see 7.21 and Figure 17).



IEC 1526/06

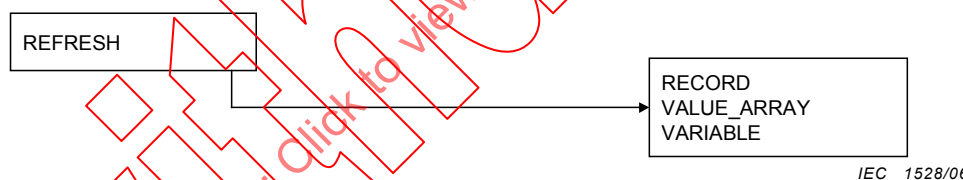
Figure 17 – RECORD

- **REFERENCE_ARRAY** is a set of EDD items of the same EDDL item type (for example, VARIABLE or MENU). An item can be referenced in the EDD via the REFERENCE_ARRAY name and the index associated with the item (see 7.22 and Figure 18).

**Figure 18 – REFERENCE_ARRAY**

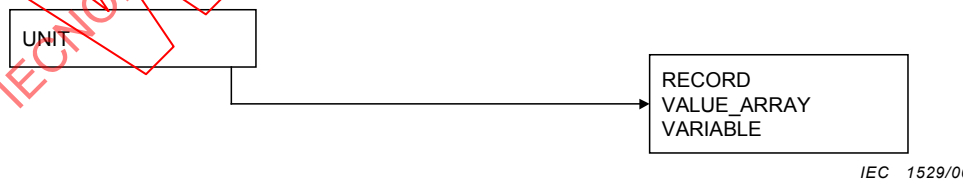
IEC 1527/06

- **Relation:** three types of relations are specified.
 - **REFRESH** describes relationships between **VARIABLES**, **RECORDS**, and **VALUE_ARRAYS** (see 7.23.2 and Figure 19).

**Figure 19 – REFRESH**

IEC 1528/06

- **UNIT** describes relationships between **VARIABLES**, **RECORDS**, and **VALUE_ARRAYS** (see 7.23.3 and Figure 20).

**Figure 20 – UNIT**

IEC 1529/06

- **WRITE_AS_ONE** RELATION describes relationships between **VARIABLES**, **RECORDS**, and **VALUE_ARRAYS** (see 7.23.4 and Figure 21).

**Figure 21 – WRITE_AS_ONE**

IEC 1530/06

- **RESPONSE_CODES** specifies the application specific response codes for use when managing a VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST, PROGRAM, or DOMAIN (see 7.24).
- **SOURCE** (see 7.25 and Figure 22).

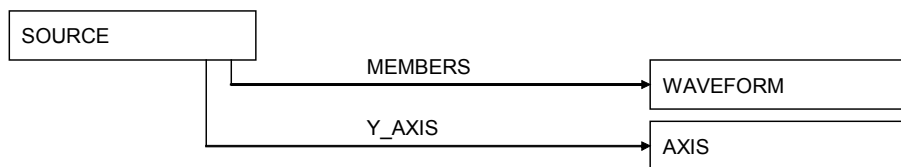


Figure 22 – SOURCE

IEC 1531/06

- **VALUE_ARRAY** describes logical groups of EDDL variables (see 7.26 and Figure 23).

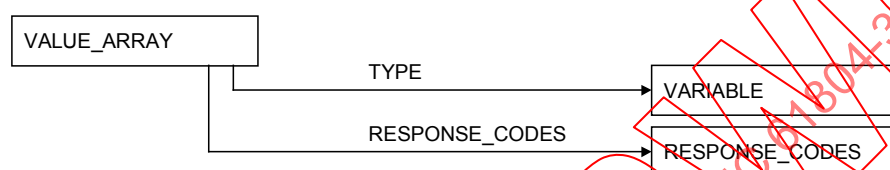


Figure 23 – VALUE_ARRAY

IEC 1532/06

- **VARIABLE** describes the data contained in the device (see 7.27 and Figure 24).

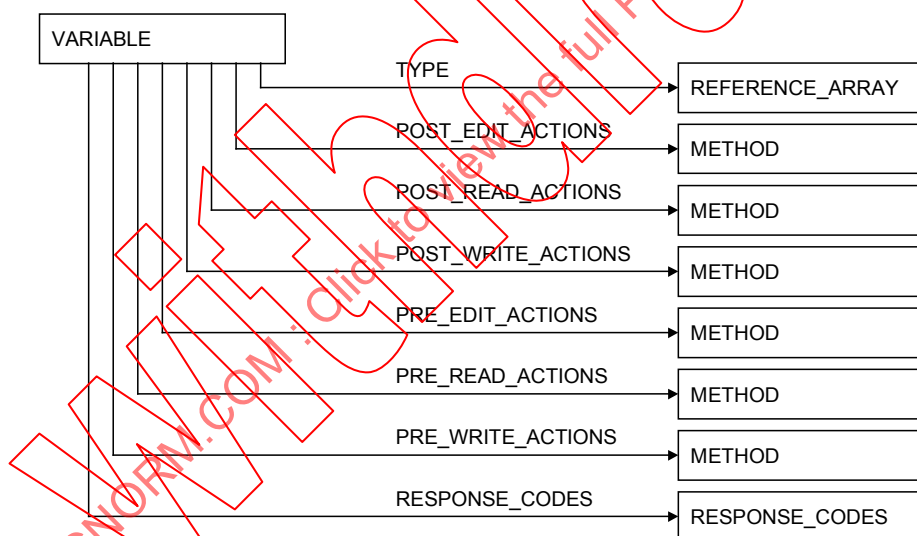


Figure 24 – VARIABLE

IEC 1533/06

- **VARIABLE_LIST** describes logical groupings of data contained in the device that may be communicated as a list (see 7.28 and Figure 25).

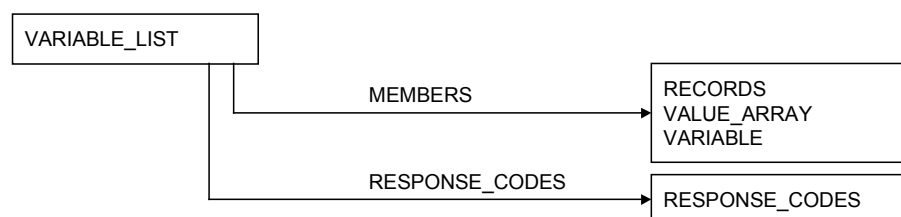
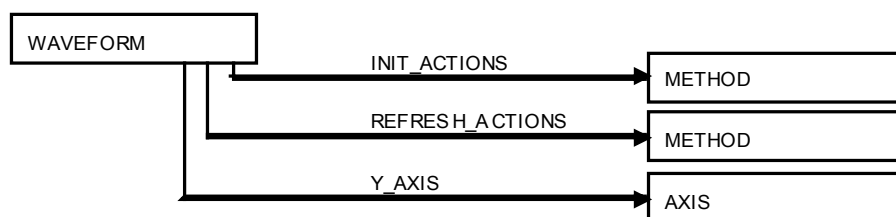


Figure 25 – VARIABLE_LIST

IEC 1534/06

- **WAVEFORM** (see 7.29 and Figure 26).



IEC 1535/06

Figure 26 – WAVEFORM

7.1.5 Common attributes

The following common attributes are used in many basic constructs:

- DEFINITION
- MEMBERS
- LABEL
- HELP
- RESPONSE_CODES

7.1.6 Special elements

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

- **Output Redirection**
 - **OPEN** is used to create a separate output file for an EDD.
 - **CLOSE** is used to close an opened output file for an EDD.
- **Conditional Expressions** specify attributes values, which are dependent on the values of variables at run-time
- **Reference** is used throughout a device description to refer to other items within the same EDDL
- **Expression** is a logical or mathematical operation on variable values in order to modify attribute values of a basic construct at run-time

7.1.7 Rules for instances

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

7.1.8 Rules for list of VARIABLES

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

- BLOCK_A PARAMETERS
- COLLECTIONS

- RECORDS
- REFERENCE_ARRAY
- VALUE_ARRAY
- VARIABLE_LISTS

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

7.2 EDD identification information

7.2.1 General structure

Purpose

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

This identification information shall be the first entry in every EDD and shall appear only once.

Lexical structure

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

NOTE 1 EDD identification information is not structured in the same way as other EDDL constructs.

NOTE 2 Strict order of the keywords of the EDD identification information is not necessary.

7.2.2 Specific attributes

7.2.2.1 DD_REVISION

Purpose

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

NOTE 1 The manufacturer should change the revision number each time a new device description is issued for a device.

NOTE 2 There can be multiple EDDs for a particular revision of a particular device, in which case the DD_REVISION distinguishes them.

Lexical structure

DD_REVISION integer

The attribute of DD_REVISION is specified in Table 2.

Table 2 – DD_REVISION attribute

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

7.2.2.2 DEVICE_REVISION

Purpose

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

Lexical structure

DEVICE_REVISION integer

The attribute of DEVICE_REVISION is specified in Table 3.

Table 3 – DEVICE_REVISION attribute

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

7.2.2.3 DEVICE_TYPE**Purpose**

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

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

Lexical structure

DEVICE_TYPE (integer, identifier)

The attribute of DEVICE_TYPE is specified in Table 4.

Table 4 – DEVICE_TYPE attribute

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

7.2.2.4 EDD_PROFILE**Purpose**

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

Lexical structure

EDD_PROFILE integer

The attribute of EDD_PROFILE is specified in Table 5.

Table 5 – EDD_PROFILE attribute

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

7.2.2.5 EDD_VERSION**Purpose**

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

Lexical structure

EDD_VERSION integer

The attribute of EDD_VERSION is specified in Table 6.

Table 6 – EDD_VERSION attribute

Usage	Attribute	Description
o	integer	is a two-octet unsigned integer containing the number 1 for this first edition of the standard

7.2.2.6 MANUFACTURER

Purpose

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

Lexical structure

MANUFACTURER [(long integer, identifier)<exp>]

The attribute of MANUFACTURER is specified in Table 7.

Table 7 – MANUFACTURER attribute

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

7.2.2.7 MANUFACTURER_EXT

Purpose

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

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

Lexical structure

MANUFACTURER_EXT string

The attribute of MANUFACTURER_EXT is specified in Table 8.

Table 8 – MANUFACTURER_EXT attribute

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

7.3 AXIS

7.3.1 General structure

Purpose

AXIS describes an axis of a CHART or GRAPH.

Lexical structure

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

The attributes of AXIS are specified in Table 9.

Table 9 – AXIS attributes

Usage	Attribute	Description
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)
o	MAX_VALUE	lexical element (see 7.3.2.1)
o	MIN_VALUE	lexical element (see 7.3.2.2)
o	SCALING	lexical element (see 7.3.2.3)
o	CONSTANT_UNIT	lexical element (see 7.3.2.4)

7.3.2 Specific attributes

7.3.2.1 MAX_VALUE

Purpose

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

Lexical structure

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

The attributes of MAX_VALUE are specified in Table 10.

Table 10 – MAX_VALUE, MIN_VALUE attribute

Usage	Attribute	Description
o	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
o	double	is a double-precision floating-point constant
o	float	is a single-precision floating-point constant
o	integer	is an integer constant
o	unsigned_integer	is an unsigned integer constant

7.3.2.2 MIN_VALUE

Purpose

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

Lexical structure

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

The attributes of MIN_VALUE are specified in Table 10.

7.3.2.3 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.

Lexical structure

SCALING (LINEAR, LOGARITHMIC)<exp>

The attributes of SCALING are specified in Table 11.

Table 11 – SCALING attribute

Usage	Attribute	Description
s	LINEAR	specifies that the tick marks on the axis should be scaled linearly
s	LOGARITHMIC	specifies that the tick marks on the axis should scaled logarithmically (base 10)

7.3.2.4 CONSTANT_UNIT

Purpose

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

Lexical structure

CONSTANT_UNIT (string)<exp>

The attribute of CONSTANT_UNIT is specified in Table 150.

7.4 BLOCK

7.4.1 BLOCK_A

7.4.1.1 General structure

Purpose

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

NOTE This BLOCK_A approach is optimized for access efficiency.

Lexical structure

BLOCK_A identifier [CHARACTERISTICS, LABEL, PARAMETERS, AXIS_ITEMS, CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISPLAY_ITEMS, FILE_ITEMS, GRAPH_ITEMS, GRID_ITEMS, IMAGE_ITEMS, LIST_ITEMS, HELP, MENU_ITEMS, METHOD_ITEMS, PARAMETER_LISTS, REFERENCE_ARRAY_ITEMS, REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS, WRITE_AS_ONE_ITEMS]

The attributes of BLOCK_A are specified in Table 12.

Table 12 – BLOCK_A attributes

Usage	Attribute	Description
m	CHARACTERISTICS	lexical element (see 7.4.1.2.1)
m	LABEL	lexical element (see 7.30.3)
m	PARAMETERS	lexical element (see 7.4.1.2.2)
o	AXIS_ITEMS	lexical element (see 7.4.1.2.3)
o	CHART_ITEMS	lexical element (see 7.4.1.2.4)
o	COLLECTION_ITEMS	lexical element (see 7.4.1.2.5)
o	EDIT_DISPLAY_ITEMS	lexical element (see 7.4.1.2.6)
o	FILE_ITEMS	lexical element (see 7.4.1.2.7)
o	GRAPH_ITEMS	lexical element (see 7.4.1.2.8)
o	GRID_ITEMS	lexical elements (see 7.4.1.2.9)
o	IMAGE_ITEMS	lexical elements (see 7.4.1.2.10)
o	LIST_ITEMS	lexical element (see 7.4.1.2.11)
o	LOCAL_PARAMETERS	lexical element (see 7.4.1.2.12)
o	HELP	lexical element (see 7.30.2)
o	MENU_ITEMS	lexical element (see 7.4.1.2.13)
o	METHOD_ITEMS	lexical element (see 7.4.1.2.14)
o	PARAMETER_LISTS	lexical element (see 7.4.1.2.15)
o	REFERENCE_ARRAY_ITEMS	lexical element (see 7.4.1.2.16)
o	REFRESH_ITEMS	lexical element (see 7.4.1.2.17)
o	SOURCE_ITEMS	lexical element (see 7.4.1.2.18)
o	UNIT_ITEMS	lexical element (see 7.4.1.2.19)
o	WAVEFORM_ITEMS	lexical element (see 7.4.1.2.20)
o	WRITE_AS_ONE_ITEMS	lexical element (see 7.4.1.2.21)

7.4.1.2 Specific attributes

7.4.1.2.1 CHARACTERISTICS

Purpose

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

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

Lexical structure

CHARACTERISTICS reference

The attribute of CHARACTERISTICS is specified in Table 13.

Table 13 – CHARACTERISTIC attribute

Usage	Attribute	Description
m	reference	is a reference to a RECORD instance

7.4.1.2.2 PARAMETERS

Purpose

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

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

Lexical structure

PARAMETERS [identifier, reference, description, help]+

The attributes of PARAMETERS are specified in Table 14.

Table 14 – PARAMETER attributes

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

7.4.1.2.3 AXIS_ITEMS

Purpose

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

Lexical structure

AXIS_ITEMS reference+

The attribute of AXIS_ITEMS is specified in Table 15.

Table 15 – AXIS_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a AXIS instance

7.4.1.2.4 CHART_ITEMS

Purpose

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

Lexical structure

CHART_ITEMS reference+

The attribute of CHART_ITEMS is specified in Table 16.

Table 16 – CHART_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a CHART instance

7.4.1.2.5 COLLECTION_ITEMS

Purpose

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

Lexical structure

COLLECTION_ITEMS reference+

The attribute of COLLECTION_ITEMS is specified in Table 17.

Table 17 – COLLECTION_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a COLLECTION instance

7.4.1.2.6 EDIT_DISPLAY_ITEMS

Purpose

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

Lexical structure

EDIT_DISPLAY_ITEMS reference+

The attribute of EDIT_DISPLAY_ITEMS is specified in Table 18.

Table 18 – EDIT_DISPLAY_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to an EDIT_DISPLAY instance

7.4.1.2.7 FILE_ITEMS

Purpose

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

Lexical structure

FILE_ITEMS reference+

The attribute of FILE_ITEMS is specified in Table 19.

Table 19 – FILE_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a FILE instance

7.4.1.2.8 GRAPH_ITEMS

Purpose

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

Lexical structure

GRAPH_ITEMS reference+

The attribute of GRAPH_ITEMS is specified in Table 20.

Table 20 – GRAPH_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a GRAPH instance

7.4.1.2.9 GRID_ITEMS

Purpose

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

Lexical structure

GRID_ITEMS reference+

The attribute of GRID_ITEMS is specified in Table 21.

Table 21 – GRID_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a GRID instance

7.4.1.2.10 IMAGE_ITEMS

Purpose

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

Lexical structure

IMAGE_ITEMS reference+

The attribute of IMAGE_ITEMS is specified in Table 22.

Table 22 – IMAGE_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a IMAGE instance

7.4.1.2.11 LIST_ITEMS

Purpose

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

Lexical structure

LIST_ITEMS reference+

The attribute of LIST_ITEMS is specified in Table 23.

Table 23 – LIST_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a LIST instance

7.4.1.2.12 LOCAL_PARAMETERS

Purpose

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

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

Lexical structure

LOCAL_PARAMETERS [identifier, reference, description, help] +

The attributes of LOCAL_PARAMETERS are specified in Table 14.

7.4.1.2.13 MENU_ITEMS

Purpose

The MENU_ITEMS list specifies the MENUs associated with BLOCK_A.

Lexical structure

MENU_ITEMS reference +

The attribute of MENU_ITEMS is specified in Table 24.

Table 24 – MENU_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a MENU instance

7.4.1.2.14 METHOD_ITEMS

Purpose

The METHOD_ITEMS list specifies the METHODS associated with BLOCK_A.

Lexical structure

METHOD_ITEMS reference +

The attribute of METHOD_ITEMS is specified in Table 25.

Table 25 – METHOD_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a METHOD instance

7.4.1.2.15 PARAMETER_LISTS

Purpose

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

Lexical structure

PARAMETER_LISTS [identifier, reference, description, help] +

The attributes of PARAMETER_LISTS are specified in Table 26.

Table 26 – PARAMETER_LISTS attributes

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

7.4.1.2.16 REFERENCE_ARRAY_ITEMS

Purpose

The REFERENCE_ARRAY_ITEMS list specifies the REFERENCE_ARRAYs associated with the BLOCK_A.

Lexical structure

REFERENCE_ARRAY_ITEMS reference+

The attribute of REFERENCE_ARRAY_ITEMS is specified in Table 27.

Table 27 – REFERENCE_ARRAY_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a REFERENCE_ARRAY instance

7.4.1.2.17 REFRESH_ITEMS

Purpose

The REFRESH_ITEMS list specify the REFRESH relations associated with the BLOCK_A.

Lexical structure

REFRESH_ITEMS reference+

The attribute of REFRESH_ITEMS is specified in Table 28.

Table 28 – REFRESH_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a REFRESH instance

7.4.1.2.18 SOURCE_ITEMS

Purpose

The SOURCE_ITEMS list specifies the SOURCES associated with the BLOCK_A.

Lexical structure

SOURCE_ITEMS reference+

The attribute of SOURCE_ITEMS is specified in Table 29.

Table 29 – SOURCE_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a SOURCE instance

7.4.1.2.19 UNIT_ITEMS**Purpose**

The UNIT_ITEMS list specify the UNIT relations associated with the BLOCK_A.

Lexical structure

UNIT_ITEMS reference+

The attribute of UNIT_ITEMS is specified in Table 30.

Table 30 – UNIT_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to an UNIT instance

7.4.1.2.20 WAVEFORM_ITEMS**Purpose**

The WAVEFORM_ITEMS list specifies the WAVEFORMs associated with the BLOCK_A.

Lexical structure

WAVEFORM_ITEMS reference+

The attribute of WAVEFORM_ITEMS is specified in Table 31.

Table 31 – WAVEFORM_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a WAVEFORM instance

7.4.1.2.21 WRITE_AS_ONE_ITEMS**Purpose**

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

Lexical structure

WRITE_AS_ONE_ITEMS reference+

The attribute of WRITE_AS_ONE_ITEMS is specified in Table 32.

Table 32 – WRITE_AS_ONE_ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a WRITE_AS_ONE instance

7.4.2 BLOCK_B

7.4.2.1 General structure

Purpose

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

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

NOTE 1 This BLOCK_B approach is optimized for memory efficiency.

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

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

Lexical structure

BLOCK_B identifier [NUMBER, TYPE]

The attributes of BLOCK_B are specified in Table 33.

Table 33 – BLOCK_B attributes

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

7.4.2.2 Specific attributes

7.4.2.2.1 NUMBER

Purpose

A device may contain several BLOCK_B instances, which are identified with the NUMBER attribute. The NUMBER attribute enumerates each BLOCK_B of the same TYPE in the device.

Lexical structure

NUMBER (integer, expression) <exp>

The attribute of NUMBER is specified in Table 34.

Table 34 – NUMBER attributes

Usage	Attribute	Description
s	integer	is the number of the BLOCK_B instance of the same TYPE
s	expression	is an expression which shall be evaluated to a non-negative integer that is within the index range of the NUMBER attribute (for the description of expressions, see 7.34.7)

7.4.2.2.2 TYPE

Purpose

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

Lexical structure

TYPE [PHYSICAL, TRANSDUCER, FUNCTION]

The attributes of TYPE are specified in Table 35.

Table 35 – TYPE attributes

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

7.5 CHART

7.5.1 General structure

Purpose

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

Lexical structure

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

The attributes of CHART are specified in Table 36.

Table 36 – CHART attributes

Usage	Attribute	Description
M	MEMBERS	lexical element (see 7.30.4)
o	CYCLE_TIME	lexical element (see 7.5.2.1)
o	HEIGHT	lexical element (see 7.5.2.2)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)
o	LENGTH	lexical element (see 7.5.2.4)
o	TYPE	lexical element (see 7.5.2.5)
o	WIDTH	lexical element (see 7.5.2.3)

7.5.2 Specific attributes

7.5.2.1 CYCLE_TIME

Purpose

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

Lexical structure

CYCLE_TIME (integer)<exp>

The attribute of CYCLE_TIME is specified in Table 37.

Table 37 – CYCLE_TIME attribute

Usage	Attribute	Description
m	integer	specifies the length of time, in ms, between successive reads of the source values in the device

7.5.2.2 HEIGHT

Purpose

HEIGHT attribute specifies how large the vertical size of the CHART should be when it is displayed. By default, the size of a CHART without a HEIGHT attribute is MEDIUM.

Lexical structure

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE)<exp>

The attribute of HEIGHT are specified in Table 38.

Table 38 – HEIGHT/WIDTH attribute

Usage	Attribute	Description
s	XX_SMALL	specifies that the CHART should occupy an extra, extra small portion of the window in which the CHART is being displayed
s	X_SMALL	specifies that the CHART should occupy an extra small portion of the window in which the CHART is being displayed
s	SMALL	specifies that the CHART should occupy a small portion of the window in which the CHART is being displayed.
s	MEDIUM	specifies that the CHART should occupy a medium portion of the window in which the CHART is being displayed
s	LARGE	specifies that the CHART should occupy a large portion of the window in which the CHART is being displayed
s	X_LARGE	specifies that the CHART should occupy an extra large portion of the window in which the CHART is being displayed
s	XX_LARGE	specifies that the CHART should occupy an extra, extra large portion of the window in which the CHART is being displayed

7.5.2.3 WIDTH

Purpose

The WIDTH attribute specifies how large the horizontal size of the CHART should be when it is displayed. By default, the size of a CHART without a HEIGHT attribute is MEDIUM.

Lexical structure

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

The attributes of WIDTH are specified in Table 38.

7.5.2.4 LENGTH**Purpose**

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

Lexical structure

LENGTH (integer) <exp>

The attribute of LENGTH is specified in Table 39.

Table 39 – LENGTH attribute

Usage	Attribute	Description
m	integer	specifies the length of time, in ms, that is displayed by the CHART

7.5.2.5 TYPE**Purpose**

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

Lexical structure

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

The attributes of TYPE are specified in Table 40.

Table 40 – TYPE attribute

Usage	Attribute	Description
s	GAUGE	a single source value is displayed as a gauge, which is a graphical representation of the data similar to the fuel gauge in an automobile
s	HORIZONTAL_BAR	the source values are displayed as a horizontal bar chart
s	SCOPE	when the source values reach the end of the display area of the CHART the display area is erased and the new source values are once again displayed starting at the beginning of the display area
s	STRIP	when the source values reach the end of the display area of the CHART, the display area is scrolled with the oldest source values being removed from the display area and the newest source values being added to the display area
s	SWEEP	when the source values reach the end of the display area of the CHART, the new source values are once again 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
s	VERTICAL_BAR	the source values are displayed as a vertical bar chart

7.6 COLLECTION

7.6.1 General structure

Purpose

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

NOTE 1 COLLECTION is intended to be used by a tool, for example, to identify parameters that should be processed together.

NOTE 2 The recommended supported nesting level is two (for example, COLLECTION of COLLECTION).

Lexical structure

COLLECTION identifier, item-type, [MEMBERS, HELP, LABEL]

The attributes of COLLECTION are specified in Table 41.

Table 41 – COLLECTION attributes

Usage	Attribute	Description
o	item-type	lexical element (see 7.6.2.1)
m	MEMBERS	lexical element (see 7.30.4)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)

7.6.2 Specific attributes

7.6.2.1 item-type

Purpose

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

Lexical structure

item-type

Table 42 shows the allowed item-types:

Table 42 – item-type

Usage	item-type	Description
s	AXIS	selection of this basic construct
s	BLOCK_A	selection of this basic construct
s	BLOCK_B	selection of this basic construct
s	CHART	selection of this basic construct
s	COLLECTION	selection of this basic construct
s	COMMAND	selection of this basic construct
s	CONNECTION	selection of this basic construct
s	DOMAIN	selection of this basic construct
s	EDIT_DISPLAY	selection of this basic construct
s	FILE	selection of this basic construct

Usage	item-type	Description
s	GRAPH	selection of this basic construct
s	LIST	selection of this basic construct
s	MENU	selection of this basic construct
s	METHOD	selection of this basic construct
s	PROGRAM	selection of this basic construct
s	RECORD	selection of this basic construct
s	REFERENCE_ARRAY	selection of this basic construct
s	REFRESH	selection of this basic construct
s	RESPONSE_CODES	selection of this basic construct
s	SOURCE	selection of this basic construct
s	UNIT	selection of this basic construct
s	VALUE_ARRAY	selection of this basic construct
s	VARIABLE	selection of this basic construct
s	VARIABLE_LIST	selection of this basic construct
s	WAVEFORM	selection of this basic construct
s	WRITE_AS_ONE	selection of this basic construct

7.6.2.2 General

The COLLECTION has only one specific attribute.

7.7 COMMAND

7.7.1 General structure

Purpose

COMMAND is a construct to support mapping of the EDD communication elements to a chosen communication system. It specifies various elements required to build a communication frame. If this construct is used by an EDDL profile, every data item needing to be transmitted shall be referenced inside a COMMAND and also in the upload-/download MENU (see 7.18.2.10).

NOTE 1 The address field of a communication frame is specified using the selectable constructs BLOCK/SLOT/NUMBER together with INDEX and MODULE as applicable. If multiple EDD application process instances, or one EDD application process instance which needs an explicit reference, exist in a device, these are specified by the additional construct CONNECTION. 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.

NOTE 2 The scope of BLOCK/SLOT/NUMBER is limited to a given EDD application process instance, i.e. the same BLOCK/SLOT/NUMBER value may be used in conjunction with different CONNECTION values to identify different data items within different EDD application process instances.

Lexical structure

COMMAND identifier, [OPERATION, TRANSACTION+, INDEX, BLOCK_B, NUMBER, SLOT, CONNECTION, HEADER, MODULE, RESPONSE_CODES]

The attributes of COMMAND are specified in Table 43.

Table 43 – COMMAND attributes

Usage	Attribute	Description
m	OPERATION	lexical element (see 7.7.2.1)
m	TRANSACTION	lexical element (see 7.7.2.2)
c	INDEX	lexical element (see 7.7.2.3)
s	BLOCK_B	lexical element (see 7.7.2.4)
s	NUMBER	lexical element (see 7.7.2.5)
s	SLOT	lexical element (see 7.7.2.6)
o	CONNECTION	lexical element (see 7.7.2.7)
o	HEADER	lexical element (see 7.7.2.8)
o	MODULE	lexical element (see 7.7.2.9)
o	RESPONSE_CODES	lexical element (see 7.30.5)

NOTE 1 Response codes appearing within a transaction apply only to that transaction.

NOTE 2 Response codes appearing outside of any transaction apply to all transactions.

NOTE 3 If the same response code is specified both inside and outside of a transaction, the response code specified within the transaction takes precedence, but only for that transaction.

7.7.2 Specific attributes

7.7.2.1 OPERATION

Purpose

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

Lexical structure

OPERATION [(COMMAND, DATA_EXCHANGE, READ, string, WRITE)<exp>]

The attributes of OPERATION are specified in Table 44.

Table 44 – OPERATION attribute

Usage	Attribute	Description
s	COMMAND	specifies that the device performs a predefined set of actions (for example, self-test, master reset). The actions to be performed are device-specific and are identified by the NUMBER attribute
s	DATA_EXCHANGE	specifies a bi-directional transaction. The input data is specified in the REPLY attribute and the output data is specified in the REQUEST attribute of the TRANSACTION
s	READ	specifies a read transaction. The field device returns the current values of the specified variables. VARIABLES shall be defined in the REPLY section of TRANSACTION.
s	string	specifies manufacturer specific services for transactions. The transaction to be performed should be explained in a corresponding specification
s	WRITE	specifies a write transaction. The field device receives the current values of the specified variables. VARIABLES shall be defined in the REQUEST section of TRANSACTION

7.7.2.2 TRANSACTION

7.7.2.2.1 General structure

Purpose

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

EXAMPLE Block commands, such as commands 4 and 5 of HART³ Universal Command Revision 4 or later, are examples of multiple transaction commands.

Lexical structure

TRANSACTION [REPLY, REQUEST, integer, RESPONSE_CODES]

The attributes of TRANSACTION are specified in Table 45.

Table 45 – TRANSACTION attributes

Usage	Attribute	Description
m	REPLY	lexical element (see 7.7.2.2.2.1)
m	REQUEST	lexical element (see 7.7.2.2.2.2)
o	integer	specifies the number of the TRANSACTIONS (if more than one TRANSACTION is used)
o	RESPONSE_CODES	lexical element (see 7.24)

7.7.2.2.2 Specific attributes

7.7.2.2.2.1 REPLY

Purpose

The REPLY attribute contains a list of data items (constants or references to variables), which are received from the device. The order of the list shall be maintained in the corresponding data field in the communicated PDU.

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

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

The attribute item-mask

NOTE 1 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.

NOTE 2 Examples are given in Annex C.

Lexical structure

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

The attributes of REPLY are specified in Table 46.

³ See four PACKED_ASCII characters to

Table 46 – REPLY and REQUEST attributes

Usage	Attribute	Description
m	reference	is a reference to a VARIABLE instance
o	item-mask	specifies a bit pattern to extract the VARIABLE from the data field The item-mask shall be used only for the data types INTEGER, UNSIGNED INTEGER, ENUMERATED and BIT_ENUMERATED. If the least significant bit is set in the item-mask, the following item in the REQUEST or REPLY list is to be used The length of the item-mask shall be 1 byte to 4 bytes and shall correspond to the length of the data item in the REQUEST or REPLY list
o	INFO	specifies that the VARIABLE is not actually stored in the device. It provides additional information to ensure correct processing
o	INDEX	specifies that the VARIABLE is used in the REQUEST or REPLY as an index into an REFERENCE_ARRAY

NOTE 3 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 the index variables are not stored in the device.

NOTE 4 INFO may be used to send or read a VARIABLE with units differing from those used in the device.

7.7.2.2.2 REQUEST

Purpose

The REQUEST attribute contains a list of data items (constants or references to variables), which are sent to the device. The order of the list shall be maintained in the corresponding data field in the communicated PDU.

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

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

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

NOTE 1 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.

NOTE 2 Examples are given in Annex C.

Lexical structure

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

The attributes of REQUEST are specified in Table 46.

7.7.2.3 INDEX

Purpose

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

Lexical structure

INDEX (integer, reference, expression)<exp>

The attributes of INDEX are specified in Table 47.

Table 47 – INDEX attribute

Usage	Attribute	Description
s	integer	is the number of the index
s	reference	is a reference to a VARIABLE instance containing the index
s	expression	is an expression which shall be evaluated to a non-negative integer that is within the index range of the INDEX attribute (for the description of expressions, see 7.34.7).

7.7.2.4 BLOCK_B**Purpose**

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

Lexical structure

BLOCK_B (reference)<exp>

The attribute of BLOCK_B is specified in Table 48.

Table 48 – BLOCK_B attribute

Usage	Attribute	Description
m	reference	is a reference to a BLOCK_B instance

7.7.2.5 NUMBER**Purpose**

The NUMBER attribute is used to enumerate the COMMANDS.

Lexical structure

NUMBER (integer, reference, expression)<exp>

The attribute of NUMBER is specified in Table 49.

Table 49 – NUMBER attribute

Usage	Attribute	Description
s	integer	is the number of the COMMAND
s	reference	is a reference to a VARIABLE instance containing the number of the COMMAND
s	expression	is an expression which shall be evaluated to a non-negative integer that is within the index range of the NUMBER attribute (for the description of expressions, see 7.34.7).

7.7.2.6 SLOT**Purpose**

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

Lexical structure

SLOT (integer, reference, expression)<exp>

The attributes of SLOT are specified in Table 50.

Table 50 – SLOT attribute

Usage	Attribute	Description
s	integer	is the number of the slot
s	reference	is a reference to a VARIABLE instance containing the slot number
s	expression	is an expression, which shall be evaluated to a non-negative integer that is within the index range of the SLOT attribute (for the description of expressions, see 7.34.7).

7.7.2.7 CONNECTION

Purpose

The CONNECTION attribute provides a reference to the basic construct CONNECTION which contains the number of an EDD application process instance.

Lexical structure

CONNECTION (reference)<exp>

The attribute of CONNECTION is specified in Table 51.

Table 51 – CONNECTION attribute

Usage	Attribute	Description
m	reference	is a reference to a CONNECTION instance specifying the EDD application

7.7.2.8 HEADER

Purpose

The HEADER attribute is provided for specific communication protocols.

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

Lexical structure

HEADER (string)<exp>

The attribute of HEADER is specified in Table 52.

Table 52 – HEADER attribute

Usage	Attribute	Description
m	string	specifies a string that is used for the communication with specific communication protocols

7.7.2.9 MODULE

Purpose

The MODULE attribute provides an external reference. Use of this attribute is out of the scope of this standard.

NOTE The MODULE is included for compatibility reasons with existing devices that use other description files.

Lexical structure

MODULE (reference) <exp>

The attribute of MODULE is specified in Table 53.

Table 53 – MODULE attribute

Usage	Attribute	Description
m	reference	is a reference to a MODULE instance in an external description file

7.8 CONNECTION

7.8.1 General structure

Purpose

A device may include multiple EDD applications. Each EDD application is represented as an EDD application process instance. The basic construct CONNECTION provides a reference to an individual EDD application process instance within a device.

The name associated with a CONNECTION construct may be used within the COMMAND attribute CONNECTION.

Lexical structure

CONNECTION identifier, APPINSTANCE

The attribute of CONNECTION is specified in Table 54.

Table 54 – CONNECTION attribute

Usage	Attribute	Description
m	APPINSTANCE	lexical element (see 7.8.2)

7.8.2 Specific attribute – APPINSTANCE

Purpose

It is possible to define multiple EDD applications in a device. Each EDD application represents an EDD application process instance. EDD application process instances allow different access levels.

Lexical structure

APPINSTANCE (integer, reference) <exp>

The attribute of APPINSTANCE is specified in Table 55.

Table 55 – APPINSTANCE attribute

Usage	Attribute	Description
s	integer	is the number of the EDD application process instance
s	reference	is a reference to a VARIABLE instance containing the number of the EDD application process instances

7.9 DOMAIN

7.9.1 General structure

Purpose

A DOMAIN represents a memory space in a device. It may contain programs or data. The identifier specifies the corresponding DOMAIN in the device. The data, programs (which are allocated in the DOMAIN), size, access, etc. are specified outside of the EDD.

NOTE 1 DOMAIN can be used to transfer large amounts of data to and from a device.

NOTE 2 To allow DOMAIN upload or download the device needs to support DOMAIN specific services.

Lexical structure

DOMAIN identifier [HANDLING, RESPONSE_CODE]

The attributes of DOMAIN are specified in Table 56.

Table 56 – DOMAIN attributes

Usage	Attribute	Description
o	HANDLING	lexical element (see 7.9.2.1)
o	RESPONSE_CODES	lexical element (see 7.30.5)

7.9.2 Specific attributes

7.9.2.1 HANDLING

Purpose

The HANDLING attribute specifies the operations which can be performed on the DOMAIN. By default, a DOMAIN without a HANDLING attribute can be read and written.

Lexical structure

HANDLING (READ, READ_WRITE, WRITE) <exp>

The attributes of HANDLING are specified in Table 57.

Table 57 – HANDLING attribute

Usage	Attribute	Description
s	READ	indicates that the DOMAIN memory contents can only be read from the device
s	READ_WRITE	indicates that the DOMAIN memory contents can be read from and written to the device
s	WRITE	indicates that the DOMAIN memory contents can only be written to the device

7.9.2.2 General

The DOMAIN has only one specific attribute.

7.10 EDIT_DISPLAY

7.10.1 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.

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

Lexical structure

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

The attributes of EDIT_DISPLAY are specified in Table 58.

Table 58 – EDIT_DISPLAY attributes

Usage	Attribute	Description
M	EDIT_ITEMS	lexical element (see 7.10.2.1)
M	LABEL	lexical element (see 7.30.3)
O	DISPLAY_ITEMS	lexical element (see 7.10.2.2)
O	POST_EDIT_ACTIONS	lexical element (see 7.10.2.3)
O	PRE_EDIT_ACTIONS	lexical element (see 7.10.2.4)

7.10.2 Specific attributes

7.10.2.1 EDIT_ITEMS

Purpose

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

NOTE 1 Specific details for presentation and editing of a data item (for example, default value, scaling factor, range) are defined within the definitions of each referenced individual item.

NOTE 2 If a WRITE_AS_ONE relation appears as an EDIT_ITEM, the user should examine all the variables in the WRITE_AS_ONE relation. The user need not modify all the values but should at least validate that the current values are acceptable.

Lexical structure

EDIT_ITEMS [(reference)<exp>]+

The attribute of EDIT_ITEMS is specified in Table 59.

Table 59 – EDIT_ITEMS attribute

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

7.10.2.2 DISPLAY_ITEMS

Purpose

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

Lexical structure

DISPLAY_ITEMS [(reference)<exp>]+

The attribute of DISPLAY_ITEM is specified in Table 60.

Table 60 – DISPLAY_ITEM attributes

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

7.10.2.3 POST_EDIT_ACTIONS

Purpose

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

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

Lexical structure

POST_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of POST_EDIT_ACTIONS are specified in Table 61.

7.10.2.4 PRE_EDIT_ACTIONS

Purpose

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

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

Lexical structure

PRE_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of PRE_EDIT_ACTIONS are specified in Table 61.

Table 61 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS attribute

Usage	Attribute	Description
m	reference	is a reference to a METHOD instance
o	DEFINITION	lexical element (see 7.30.1)

7.11 FILE

7.11.1 General structure

Purpose

FILE describes an area of persistent storage that is maintained by the EDD application. Each physical device has its own copy of the store, i.e., a FILE associated with an instance of a device, not with the type of a device. The MEMBERS attribute defines the structure of the store.

Lexical structure

FILE identifier [MEMBERS, HELP, LABEL]

The attributes of FILE are specified in Table 62.

Table 62 – FILE attributes

Usage	Attribute	Description
m	MEMBERS	lexical element (see 7.30.4)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)

7.11.2 General

FILE has only one specific attribute.

7.12 GRAPH

7.12.1 General structure

Purpose

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

Lexical structure

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

The attributes of GRAPH are specified in Table 63.

Table 63 – GRAPH attributes

Usage	Attribute	Description
m	MEMBERS	lexical element (see 7.30.4)
o	HEIGHT	lexical element (see 7.12.2.1)
o	WIDTH	lexical element (see 7.12.2.2)
o	CYCLE_TIME	lexical element (see 7.12.2.3)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)
o	X_AXIS	lexical element (see 7.12.2.4)

7.12.2 Specific attributes

7.12.2.1 HEIGHT

Purpose

The HEIGHT attribute specifies how large the vertical size of the GRAPH should be when it is displayed. By default, the size of a GRAPH without a HEIGHT attribute is MEDIUM.

Lexical structure

HEIGHT (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE)<exp>

The attribute of HEIGHT are specified in Table 64.

Table 64 – HEIGHT/WIDTH attributes

Usage	Attribute	Description
s	XX_SMALL	specifies that the GRAPH should occupy an extra, extra small portion of the window in which the GRAPH is being displayed
s	X_SMALL	specifies that the GRAPH should occupy an extra small portion of the window in which the GRAPH is being displayed
s	SMALL	specifies that the GRAPH should occupy a small portion of the window in which the GRAPH is being displayed.
s	MEDIUM	specifies that the GRAPH should occupy a medium portion of the window in which the GRAPH is being displayed
s	LARGE	specifies that the GRAPH should occupy a large portion of the window in which the GRAPH is being displayed
s	X_LARGE	specifies that the GRAPH should occupy an extra large portion of the window in which the GRAPH is being displayed
s	XX_LARGE	specifies that the GRAPH should occupy an extra, extra large portion of the window in which the GRAPH is being displayed

7.12.2.2 WIDTH

Purpose

The WIDTH attribute specifies how large the horizontal size of the GRAPH should be when it is displayed. By default, the size of a GRAPH without a WIDTH attribute is MEDIUM.

Lexical structure

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE)<exp>

The attributes of WIDTH are specified in Table 64.

7.12.2.3 CYCLE_TIME

Purpose

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

Lexical structure

CYCLE_TIME (integer)<exp>

The attribute of CYCLE_TIME is specified in Table 65.

Table 65 – CYCLE_TIME attribute

Usage	Attribute	Description
m	integer	specifies the length of time, in ms, between successive reads of the source values in the device

7.12.2.4 X_AXIS

Purpose

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

Lexical structure

X_AXIS (reference)<exp>

The attribute of X_AXIS is specified in Table 66.

Table 66 – X_AXIS attribute

Usage	Attribute	Description
m	reference	is a reference to an AXIS instance

7.13 GRID

7.13.1 General structure

Purpose

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

Lexical structure

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

The attributes of GRID are specified in Table 67.

Table 67 – GRID attributes

Usage	Attribute	Description
m	VECTORS	lexical element (see 7.13.2.1)
o	HANDLING	lexical element (see 7.13.2.2)
o	HEIGHT	lexical element (see 7.13.2.3)
o	WIDTH	lexical element (see 7.13.2.4)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)
o	ORIENTATION	lexical element (see 7.13.2.5)
o	VALIDITY	lexical element (see 7.13.2.6)

7.13.2 Specific attributes

7.13.2.1 VECTORS

Purpose

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

Lexical structure

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

The attributes of VECTORS are specified in Table 68.

Table 68 – VECTORS attribute

Usage	Attribute	Description
m	description	is a string containing a short description of the data within the vector
o	reference	is a reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance, an element of a REFERENCE_ARRAY instance, a VALUE_ARRAY instance or a REFERENCE_ARRAY instance
	string	is a string constant or a reference that resolves to a string
o	double	is a double precision floating point constant
o	float	is a single precision floating point constant
o	integer	is an integer constant
o	unsigned_integer	is an unsigned integer constant

7.13.2.2 HANDLING

Purpose

The HANDLING attribute specifies the operations which can be performed on the data in the GRID. By default, the data within a GRID without a HANDLING attribute can be read and written.

Lexical structure

HANDLING (READ, READ_WRITE, WRITE)<exp>

The attributes of HANDLING are specified in Table 69.

Table 69 – HANDLING attribute

Usage	Attribute	Description
s	READ	specifies that the data within the GRID may be read
s	READ_WRITE	specifies that the data within the GRID may be read and written
s	WRITE	specifies that the data within the GRID may be written

7.13.2.3 HEIGHT

Purpose

The HEIGHT attribute specifies how large the vertical size of the GRID should be when it is displayed. By default, the size of a GRID without a HEIGHT attribute is MEDIUM.

Lexical structure

HEIGHT (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

The attributes of HEIGHT are specified in Table 70.

Table 70 – HEIGHT attribute

Usage	Attribute	Description
s	XX_SMALL	specifies that the GRID should occupy an extra, extra small portion of the window in which the GRID is being displayed
s	X_SMALL	specifies that the GRID should occupy an extra small portion of the window in which the GRID is being displayed
s	SMALL	specifies that the GRID should occupy a small portion of the window in which the GRID is being displayed.
s	MEDIUM	specifies that the GRID should occupy a medium portion of the window in which the GRID is being displayed
s	LARGE	specifies that the GRID should occupy a large portion of the window in which the GRID is being displayed
s	X_LARGE	specifies that the GRID should occupy an extra large portion of the window in which the GRID is being displayed
s	XX_LARGE	specifies that the GRID should occupy an extra, extra large portion of the window in which the GRID is being displayed

7.13.2.4 WIDTH

Purpose

The WIDTH attribute specifies how large the horizontal size of the GRID should be when it's displayed. By default, the size of a GRID without a WIDTH attribute is MEDIUM.

Lexical structure

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

The attributes of WIDTH are specified in Table 70.

7.13.2.5 ORIENTATION

Purpose

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

Lexical structure

ORIENTATION (HORIZONTAL, VERTICAL)<exp>

The attributes of ORIENTATION are specified in Table 71.

Table 71 – ORIENTATION attribute

Usage	Attribute	Description
s	HORIZONTAL	specifies that the vectors are to be displayed in rows
s	VERTICAL	specifies that the vectors are to be displayed in columns

7.13.2.6 VALIDITY

Purpose

The VALIDITY attribute specifies whether a GRID is valid or invalid. When a GRID is invalid it is not displayed. By default, a GRID without a VALIDITY attribute is always valid.

NOTE VALIDITY should be expressed using a conditional (IF, IF-ELSE, SELECT).

Lexical structure

VALIDITY [(FALSE, TRUE)<exp>]

The attributes of VALIDITY are specified in Table 72.

Table 72 – VALIDITY attributes

Usage	Attribute	Description
s	FALSE	specifies that the GRID instance is invalid
s	TRUE	specifies that the GRID instance is valid

7.14 IMAGE

7.14.1 General structure

Purpose

IMAGE describes a visual image.

Lexical structure

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

The attributes of IMAGE are specified in Table 73.

Table 73 – IMAGE attributes

Usage	Attribute	Description
m	PATH	lexical element (see 7.14.2.1)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)
o	LINK	lexical element (see 7.14.2.2)
o	VALIDITY	lexical element (see 7.14.2.3)

7.14.2 Specific attributes

7.14.2.1 PATH

Purpose

The PATH attribute specifies the filename of the image. Images for multiple locales may be specified using the same localization technique used with string literals.

EXAMPLE “|en|english.jpeg|de|german.jpeg”

Lexical structure

PATH (string)<exp>

The attributes of PATH are specified in Table 74.

Table 74 – PATH attribute

Usage	Attribute	Description
m	string	specifies the filename of the image

7.14.2.2 LINK

Purpose

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

Lexical structure

LINK (reference)<exp>

The attributes of LINK are specified in Table 75.

Table 75 – LINK attribute

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

7.14.2.3 VALIDITY

Purpose

The VALIDITY attribute specifies whether an IMAGE is valid or invalid. If an IMAGE is invalid, it should not be displayed. An IMAGE without a VALIDITY is always valid.

NOTE VALIDITY should be expressed using a conditional (IF, IF-ELSE, SELECT).

Lexical structure

VALIDITY [(FALSE, TRUE)<exp>]

The attributes of VALIDITY are specified in Table 76.

Table 76 – VALIDITY attributes

Usage	Attribute	Description
s	FALSE	specifies that the IMAGE instance is invalid
s	TRUE	specifies that the IMAGE instance is valid

7.15 IMPORT

7.15.1 General structure

Purpose

The EDDL constructs previously specified in this clause 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.

NOTE 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. 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 27 shows the EDDL import mechanisms.

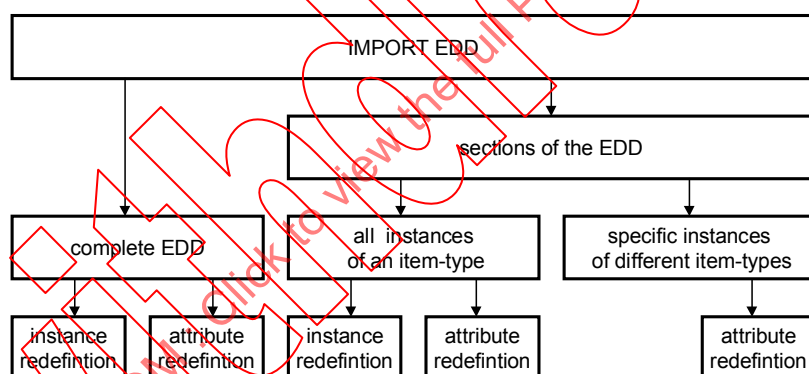


Figure 27 – EDDL import mechanisms

IEC 1536/06

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

Lexical structure

```

IMPORT ((DD_REVISION,    DEVICE_REVISION,    DEVICE_TYPE,    EDD_PROFILE,
EDD_VERSION, MANUFACTURER, MANUFACTURER_EXT), FILE_NAME),
(EVERYTHING,
(AXES, BLOCKS_A, BLOCKS_B, CHART, COLLECTIONS, COMMANDS, CONNECTIONS,
DOMAINS, EDIT_DISPLAYS, FILES, GRAPHS, LISTS, MENUS, METHODS, PROGRAMS,
RECORDS, REFERENCE_ARRAYS, REFRESHES, RELATIONS, RESPONSE_CODES, SOURCES,
UNITS, VALUE_ARRAYS, VARIABLES, VARIABLE_LISTS, WAVEFORMS, WRITE_AS_ONES,
reference*,
[DELETE identifier]*,
[REDEFINE identifier]*,
attribute-redefinition*))
    
```

The attributes are specified in Table 77.

Table 77 – Importing Device Description

Usage	Attribute	Description
o	attribute-redefinition	(see 7.15.2)
o	AXES	specifies that all AXIS instances of an external EDD are imported
o	BLOCKS_A	specifies that all BLOCK_A instances of an external EDD are imported
o	BLOCKS_B	specifies that all BLOCK_B instances of an external EDD are imported
o	CHARTS	specifies that all CHART instances of an external EDD are imported
o	COLLECTIONS	specifies that all COLLECTION instances of an external EDD are imported
o	COMMANDS	specifies that all COMMAND instances of an external EDD are imported
o	CONNECTIONS	specifies that all CONNECTION instances of an external EDD are imported
m	DD_REVISION	(see 7.2.2.1)
o	DELETE	specifies that an imported instance shall be deleted
m	DEVICE_REVISION	(see 7.2.2.2)
m	DEVICE_TYPE	(see 7.2.2.3)
o	DOMAINS	specifies that all DOMAIN instances of an external EDD are imported
o	EDD_PROFILE	(see 7.2.2.4)
o	EDD_VERSION	(see 7.2.2.5)
o	EDIT_DISPLAYS	specifies that all EDIT_DISPLAY instances of an external EDD are imported
o	EVERYTHING	specifies that all items of an external EDD are imported. If this form is used, no item types and no single instances can be selected. The attributes DELETE, REDEFINE and attribute-redefinition may be used
o	FILE_NAME	specifies a string which contains a file name and its path in a directory structure. The file contains the EDD which shall be imported. Using this kind of referencing, the specification of the EDD identification information is not necessary
o	FILES	specifies that all FILE instances of an external EDD are imported
o	GRAPHS	specifies that all GRAPH instances of an external EDD are imported
o	LISTS	specifies that all LIST instances of an external EDD are imported
m	MANUFACTURER	(see 7.2.2.6)
c	MANUFACTURER_EXT	(see 7.2.2.7)
o	MENUS	specifies that all MENU instances of an external EDD are imported
o	METHODS	specifies that all METHOD instances of an external EDD are imported
o	PROGRAMS	specifies that all PROGRAM instances of an external EDD are imported
o	RECORDS	specifies that all RECORD instances of an external EDD are imported
o	REDEFINE	specifies that all attributes of an imported instance shall be redefined
o	reference	specifies that a single instance is imported. A single instance shall not be imported, if it is already included in an item type import
o	REFERENCE_ARRAYS	specifies that all REFERENCE_ARRAY instances of an external EDD are imported
o	REFRESHES	specifies that all REFRESH instances of an external EDD are imported
o	RELATIONS	specifies that all REFRESH, UNIT and WRITE_AS_ONE instances of an external EDD are imported
o	RESPONSE_CODES	specifies that all RESPONSE_CODE instances of an external EDD are imported
o	SOURCES	specifies that all SOURCE instances of an external EDD are imported
o	UNITS	specifies that all UNIT instances of an external EDD are imported
o	VALUE_ARRAYS	specifies that all VALUE_ARRAY instances of an external EDD are imported
o	VARIABLE_LISTS	specifies that all VARIABLE_LIST instances of an external EDD are imported
o	VARIABLES	specifies that all VARIABLE instances of an external EDD are imported
o	WAVEFORMS	specifies that all WAVEFORM instances of an external EDD are imported
o	WRITE_AS_ONES	specifies that all WRITE_AS_ONE instances of an external EDD are imported

7.15.2 Specific attributes

7.15.2.1 Attribute redefinition

Purpose

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

The redefinition attributes are specified in Table 78.

Table 78 – Redefinition attributes

Usage	Attribute	Description
s	ADD	adds the following specified attribute. "A" indicates that ADD can be applied to the attribute in the redefinition rules tables (see Table 79 to Table 103)
s	DELETE	deletes the following specified attribute. "D" indicates that DELETE can be applied to the attribute in the redefinition rules tables (see Table 79 to Table 103)
s	REDEFINE	redefine the following specified attribute. "R" indicates that REDEFINE can be applied to the attribute in the redefinition rules tables (see Table 79 to Table 103)

Lexical structure – AXIS

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

The redefinition rules of AXIS are specified in Table 78 and Table 79.

Table 79 – Redefinition rules for AXIS attributes

A	D	R	Attribute	Description
	•	•	HELP	lexical element (see 7.30.2)
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	MAX_VALUE	lexical element (see 7.3.2.1)
	•	•	MIN_VALUE	lexical element (see 7.3.2.2)
	•	•	SCALING	lexical element (see 7.3.2.3)

Lexical structure – BLOCK_A

BLOCK identifier, [[ADD, DELETE, REDEFINE], [CHARACTERISTICS, LABEL, PARAMETERS, parameters-list-element, AXIS_ITEMS, CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISP_ITEMS, FILE_ITEMS, GRAPH_ITEMS, LIST_ITEMS, LOCAL_PARAMETERS, local-parameters-list-element, HELP, MENU_ITEMS, METHOD_ITEMS, PARAMETER_LISTS, parameters-lists-list-element, REFERENCE_ARRAY_ITEMS, REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS, WRITE_AS_ONE_ITEMS]]+

The redefinition rules of BLOCK_A are specified in Table 78 and Table 80.

Table 80 – Redefinition rules for BLOCK_A attributes

A	D	R	Attribute	Description
		•	CHARACTERISTICS	lexical element (see 7.4.1.2.1)
		•	LABEL	lexical element (see 7.30.3)
		•	PARAMETERS	lexical element (see 7.4.1.2.2)
•	•	•	parameters-list-element	single enumeration list elements of PARAMETERS may be changed
		•	AXIS_ITEMS	lexical element (see 7.4.1.2.3)
		•	CHART_ITEMS	lexical element (see 7.4.1.2.4)
		•	COLLECTION_ITEMS	lexical element (see 7.4.1.2.5)
		•	EDIT_DISP_ITEMS	lexical element (see 7.4.1.2.6)
		•	FILE_ITEMS	lexical element (see 7.4.1.2.7)
		•	GRAPH_ITEMS	lexical element (see 7.4.1.2.8)
		•	GRID_ITEMS	lexical elements (see 7.4.1.2.9)
		•	IMAGE_ITEMS	lexical elements (see 7.4.1.2.10)
		•	LIST_ITEMS	lexical element (see 7.4.1.2.11)
		•	LOCAL_PARAMETERS	lexical element (see 7.4.1.2.12)
•	•	•	local-parameters-list-element	single enumeration list elements of LOCAL_PARAMETERS may be changed
		•	HELP	lexical element (see 7.30.2)
		•	MENU_ITEMS	lexical element (see 7.4.1.2.13)
		•	METHOD_ITEMS	lexical element (see 7.4.1.2.14)
		•	PARAMETER_LISTS	lexical element (see 7.4.1.2.15)
•	•	•	parameters-lists-list-element	single enumeration list elements of PARAMETERS_LISTS may be changed
		•	REFERENCE_ARRAY_ITEMS	lexical element (see 7.4.1.2.16)
		•	REFRESH_ITEMS	lexical element (see 7.4.1.2.17)
		•	SOURCE_ITEMS	lexical element (see 7.4.1.2.18)
		•	UNIT_ITEMS	lexical element (see 7.4.1.2.19)
		•	WAVEFORM_ITEMS	lexical element (see 7.4.1.2.20)
		•	WRITE_AS_ONE_ITEMS	lexical element (see 7.4.1.2.21)

Lexical structure – BLOCK_B

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

The redefinition rules of BLOCK_B are specified in Table 78 and Table 81.

Table 81 – Redefinition rules for BLOCK_B attributes

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

Lexical structure — CHART

CHART identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, CYCLE_TIME, DISPLAY_SIZE, HELP, LABEL, LENGTH, TYPE]]+

The redefinition rules of COLLECTION are specified in Table 78 and Table 82.

Table 82 – Redefinition rules for CHART attributes

A	D	R	Attribute	Description
		•	MEMBERS	lexical element (see 7.30.4)
•	•	•	members-list-element	single enumeration list elements of MEMBERS may be changed
		•	CYCLE_TIME	lexical element (see 7.5.2.1)
		•	HEIGHT	lexical element (see 7.5.2.2)
		•	WIDTH	lexical element (see 7.5.2.3)
		•	HELP	lexical element (see 7.30.2)
		•	LABEL	lexical element (see 7.30.3)
		•	LENGTH	lexical element (see 7.5.2.4)
		•	TYPE	lexical element (see 7.5.2.5)

Lexical structure — COLLECTION

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

The redefinition rules of COLLECTION are specified in Table 78 and Table 83.

Table 83 – Redefinition rules for COLLECTION attributes

A	D	R	Attribute	Description
		•	MEMBERS	lexical element (see 7.30.4)
•	•	•	members-list-element	single enumeration list elements of MEMBERS may be changed
		•	HELP	lexical element (see 7.30.2)
		•	LABEL	lexical element (see 7.30.3)

Lexical structure — COMMAND

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

The redefinition rules of COMMAND are specified in Table 78 and Table 84.

Table 84 – Redefinition rules for COMMAND attributes

A	D	R	Attribute	Description
		•	OPERATION	lexical element (see 7.7.2.1)
•	•	•	TRANSACTION	lexical element (see 7.7.2.2)
		•	INDEX	lexical element (see 7.7.2.3)
		•	BLOCK_B	lexical element (see 7.7.2.4)
		•	NUMBER	lexical element (see 7.7.2.5)
		•	SLOT	lexical element (see 7.7.2.6)
		•	CONNECTION	lexical element (see 7.7.2.7)
		•	HEADER	lexical element (see 7.7.2.8)
		•	MODULE	lexical element (see 7.7.2.9)
		•	RESPONSE_CODES	lexical element (see 7.30.5)
•	•	•	response-codes-list-element	single enumeration list elements of RESPONSE_CODE may be changed

Lexical structure — CONNECTION

CONNECTION identifier, [REDEFINE, [APPINSTANCE]]+

The redefinition rules of CONNECTION are specified in Table 78 and Table 85.

Table 85 – Redefinition rules for CONNECTION attributes

A	D	R	Attribute	Description
		•	APPINSTANCE	lexical element (see 7.8.2)

Lexical structure — DOMAIN

DOMAIN identifier, [DELETE, REDEFINE], [HANDLING, RESPONSE_CODES] +

The redefinition rules of DOMAIN are specified in Table 78 and Table 86.

Table 86 – Redefinition rules for DOMAIN attributes

A	D	R	Attribute	Description
	•	•	HANDLING	lexical element (see 7.9.2)
	•	•	RESPONSE_CODES	lexical element (see 7.30.5)

Lexical structure – EDIT_DISPLAY

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 78 and Table 87.

Table 87 – Redefinition rules for EDIT_DISPLAY attributes

A	D	R	Attribute	Description
		•	EDIT_ITEMS	lexical element (see 7.10.2.1)
		•	LABEL	lexical element (see 7.30.3)
	•	•	DISPLAY_ITEMS	lexical element (see 7.10.2.2)
	•	•	POST_EDIT_ACTIONS	lexical element (see 7.10.2.3)
	•	•	PRE_EDIT_ACTIONS	lexical element (see 7.10.2.4)

Lexical structure — FILE

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

The redefinition rules of FILE are specified in Table 78 and Table 88.

Table 88 – Redefinition rules for FILE attributes

A	D	R	Attribute	Description
		•	MEMBERS	lexical element (see 7.30.4)
•	•	•	members-list-element	single enumeration list elements of MEMBERS may be changed
	•	•	HELP	lexical element (see 7.30.2)
	•	•	LABEL	lexical element (see 7.30.3)

Lexical structure — GRAPH

GRAPH identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, DISPLAY_SIZE, HELP, LABEL, Y_AXIS]]+

The redefinition rules of GRAPH are specified in Table 78 and Table 89.

Table 89 – Redefinition rules for GRAPH attributes

A	D	R	Attribute	Description
		•	MEMBERS	lexical element (see 7.30.4)
•	•	•	members-list-element	single enumeration list elements of MEMBERS may be changed
	•	•	HEIGHT	lexical element (see 7.12.2.1)
	•	•	WIDTH	lexical element (see 7.12.2.2)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	CYCLE_TIME	lexical element (see 7.12.2.3)
	•	•	X_AXIS	lexical element (see 7.12.2.4)

Lexical structure — GRID

GRID identifier, [[ADD, DELETE, REDEFINE], PATH, HELP, LABEL, LINK, VALIDITY]]+

The redefinition rules of GRAPH are specified in Table 78 and Table 90.

Table 90 – Redefinition rules for GRID attributes

A	D	R	Attribute	Description
		•	VECTORS	lexical element (see 7.13.2.1)
	•	•	HANDLING	lexical element (see 7.13.2.2)
	•	•	HEIGHT	lexical element (see 7.13.2.3)
	•	•	WIDTH	lexical element (see 7.13.2.4)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	ORIENTATION	lexical element (see 7.13.2.5)
	•	•	VALIDITY	lexical element (see 7.13.2.6)

Lexical structure — IMAGE

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

The redefinition rules of GRAPH are specified in Table 78 and Table 91.

Table 91 – Redefinition rules for IMAGE attributes

A	D	R	Attribute	Description
		•	PATH	lexical element (see 7.14.2.1)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	LINK	lexical element (see 7.14.2.2)
	•	•	VALIDITY	lexical element (see 7.14.2.3)

Lexical structure — LIST

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

The redefinition rules of LIST are specified in Table 78 and Table 92.

Table 92 – Redefinition rules for LIST attributes

A	D	R	Attribute	Description
		•	TYPE	lexical element (see 7.17.2.1)
	•	•	CAPACITY	lexical element (see 7.17.2.2)
	•	•	COUNT	lexical element (see 7.17.2.3)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	LABEL	lexical element (see 7.30.3)

Lexical structure — MENU

MENU identifier, [[ADD, DELETE, REDEFINE], [ITEMS, items-list-element, LABEL, ACCESS, ENTRY, HELP, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, PURPOSE, ROLE, STYLE, VALIDITY]]+

The redefinition rules of MENU are specified in Table 78 and Table 93.

Table 93 – Redefinition rules for MENU attributes

A	D	R	Attribute	Description
		•	ITEMS	lexical element (see 7.18.2.1)
•	•	•	items-list-element	single enumeration list elements of ITEMS be changed
		•	LABEL	lexical element (see 7.30.3)
	•	•	ACCESS	lexical element (see 7.18.2.2)
	•	•	ENTRY	lexical element (see 7.18.2.3)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	POST_EDIT_ACTIONS	lexical element (see 7.18.2.4)
	•	•	POST_READ_ACTIONS	lexical element (see 7.18.2.5)
	•	•	POST_WRITE_ACTIONS	lexical element (see 7.18.2.6)
	•	•	PRE_EDIT_ACTIONS	lexical element (see 7.18.2.7)
	•	•	PRE_READ_ACTIONS	lexical element (see 7.18.2.8)
	•	•	PRE_WRITE_ACTIONS	lexical element (see 7.18.2.9)
	•	•	PURPOSE	lexical element (see 7.18.2.10)
	•	•	ROLE	lexical element (see 7.18.2.11)
	•	•	STYLE	lexical element (see 7.18.2.12)
	•	•	VALIDITY	lexical element (see 7.18.2.13)

Lexical structure — METHOD

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

The redefinition rules of METHOD are specified in Table 78 and Table 94.

Table 94 – Redefinition rules for METHOD attributes

A	D	R	Attribute	Description
	•	•	ACCESS	lexical element (see 7.19.2.1)
		•	CLASS	lexical element (see 7.19.2.2)
		•	DEFINITION	lexical element (see 7.30.1)
		•	LABEL	lexical element (see 7.30.3)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	VALIDITY	lexical element (see 7.19.2.3)

Lexical structure — PROGRAM

PROGRAM identifier, [[DELETE, REDEFINE], [ARGUMENT, argument-list-element, RESPONSE_CODES]]+

The redefinition rules of PROGRAM are specified in Table 78 and Table 95.

Table 95 – Redefinition rules for PROGRAM attributes

A	D	R	Attribute	Description
		•	ARGUMENT	lexical element (see 7.20.2)
	•	•	argument-list-element	single enumeration list elements of ARGUMENTS may be changed
	•	•	RESPONSE_CODES	lexical element (see 7.30.5)

Lexical structure — RECORD

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

The redefinition rules of RECORD are specified in Table 78 and Table 96.

Table 96 – Redefinition rules for RECORD attributes

A	D	R	Attribute	Description
		•	MEMBERS	lexical element (see 7.30.4)
•	•	•	members-list-element	single enumeration list elements of MEMBERS may be changed
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	RESPONSE_CODES	lexical element (see 7.30.5)

Lexical structure – REFERENCE_ARRAY

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

The redefinition rules of REFERENCE_ARRAY are specified in Table 78 and Table 97.

Table 97 – Redefinition rules for REFERENCE_ARRAY attributes

A	D	R	Attribute	Description
		•	ELEMENTS	lexical element (see 7.22.2)
•	•	•	elements-list-element	single enumeration list elements of ELEMENTS may be changed
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	HELP	lexical element (see 7.30.2)

Lexical structure – RESPONSE_CODES

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

The redefinition rules of RESPONSE_CODES are specified in Table 78 and Table 98.

Table 98 – Redefinition rules for RESPONSE_CODES attributes

A	D	R	Attribute	Description
•	•	•	response-code-list-element	single enumeration list elements of RESPONSE_CODE may be changed

Lexical structure – SOURCE

SOURCE identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, EMPHASIS, HELP, LABEL, LINE_TYPE, Y_AXIS]]⁺

The redefinition rules of SOURCE are specified in Table 78 and Table 99.

Table 99 – Redefinition rules for SOURCE attributes

A	D	R	Attribute	Description
		•	MEMBERS	lexical element (see 7.30.4)
•	•	•	members-list-element	single enumeration list elements of MEMBERS may be changed
		•	EMPHASIS	lexical element (see 7.25.2.1)
		•	HELP	lexical element (see 7.30.2)
		•	LABEL	lexical element (see 7.30.3)
		•	LINE_COLOR	lexical element (see 7.25.2.2)
		•	LINE_TYPE	lexical element (see 7.25.2.3)
		•	Y_AXIS	lexical element (see 7.25.2.4)
		•	EXIT_ACTIONS	lexical element (see 7.25.2.5)
		•	INIT_ACTIONS	lexical element (see 7.25.2.6)
		•	REFRESH_ACTIONS	lexical element (see 7.25.2.7)

Lexical structure – VALUE_ARRAY

VALUE_ARRAY identifier, [[DELETE, REDEFINE], [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, RESPONSE_CODES]]⁺

The redefinition rules of VALUE_ARRAY are specified in Table 78 and Table 100.

Table 100 – Redefinition rules for VALUE_ARRAY attributes

A	D	R	Attribute	Description
		•	LABEL	lexical element (see 7.30.3)
		•	NUMBER_OF_ELEMENTS	lexical element (see 7.26.2.1)
		•	TYPE	lexical element (see 7.26.2.2)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	RESPONSE_CODES	lexical element (see 7.30.5)

Lexical structure — VARIABLE

VARIABLE identifier, [[ADD, DELETE, REDEFINE], [CLASS, LABEL, TYPE, DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, MAX_VALUE, MIN_VALUE, SCALING_FACTOR, type-list-element, CONSTANT_UNIT, HANDLING, HELP, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTION, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, READ_TIMEOUT, RESPONSE_CODES, STYLE, VALIDITY, WRITE_TIMEOUT]]+

The redefinition rules are specified in Table 78 and Table 101.

Table 101 – Redefinition rules for VARIABLE attributes

A	D	R	Attribute	Description
		•	CLASS	lexical element (see 7.27.2.1)
		•	LABEL	lexical element (see 7.30.3)
			TYPE	lexical element (see 7.27.2.2)
	•	•	DEFAULT_VALUE	lexical element (see 7.27.2.2)
	•	•	DISPLAY_FORMAT	lexical element (see 7.27.2.2)
	•	•	EDIT_FORMAT	lexical element (see 7.27.2.2)
	•	•	INITIAL_VALUE	lexical element (see 7.27.2.2)
	•	•	MAX_VALUE	lexical element (see 7.27.2.2)
	•	•	MIN_VALUE	lexical element (see 7.27.2.2)
	•	•	SCALING_FACTOR	lexical element (see 7.27.2.2)
•	•	•	type-list-element	single enumeration list elements of every TYPE may be changed
	•	•	CONSTANT_UNIT	lexical element (see 7.27.2.3)
	•	•	HANDLING	lexical element (see 7.27.2.4)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	POST_EDIT_ACTIONS	lexical element (see 7.27.2.6)
	•	•	POST_READ_ACTIONS	lexical element (see 7.27.2.8)
	•	•	POST_WRITE_ACTIONS	lexical element (see 7.27.2.9)
	•	•	PRE_EDIT_ACTION	lexical element (see 7.27.2.10)
	•	•	PRE_READ_ACTIONS	lexical element (see 7.27.2.11)
	•	•	PRE_WRITE_ACTIONS	lexical element (see 7.27.2.12)
	•	•	REFRESH_ACTIONS	lexical element (see 7.27.2.13)
	•	•	READ_TIMEOUT	lexical element (see 7.27.2.13)
	•	•	RESPONSE_CODES	lexical element (see 7.30.5)
	•	•	STYLE	lexical element (see 7.27.2.15)
	•	•	VALIDITY	lexical element (see 7.27.2.16)
	•	•	WRITE_TIMEOUT	lexical element (see 7.27.2.13)

Lexical structure – VARIABLE_LIST

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

The redefinition rules of VARIABLE_LIST are specified in Table 78 and Table 102.

Table 102 – Redefinition rules for VARIABLE_LIST attributes

A	D	R	Attribute	Description
		•	MEMBERS	lexical element (see 7.30.4)
•	•	•	members-list-element	single enumeration list elements of MEMBERS may be changed.
	•	•	HELP	lexical element (see 7.30.2)
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	RESPONSE_CODES	lexical element (see 7.30.5)

Lexical structure — WAVEFORM

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_TYPE, REFRESH_ACTIONS, Y_AXIS]]+

The redefinition rules of WAVEFORM are specified in Table 78 and Table 103.

Table 103 – Redefinition rules for WAVEFORM attributes

A	D	R	Attribute	Description
		•	TYPE	lexical element (see 7.29.2.1)
•		•	X_VALUES	lexical element (see 7.29.2.1)
•		•	Y_VALUES	lexical element (see 7.29.2.1)
	•	•	NUMBER_OF_POINTS	lexical element (see 7.29.2.1)
		•	X_INITIAL	lexical element (see 7.29.2.1)
		•	X_INCREMENT	lexical element (see 7.29.2.1)
	•	•	EMPHASIS	lexical element (see 7.29.2.2)
	•	•	EXIT_ACTIONS	lexical element (see 7.29.2.3)
	•	•	HANDLING	lexical element (see 7.29.2.3)
	•	•	HELP	lexical element (see 7.30.2)
	•	•	INIT_ACTIONS	lexical element (see 7.29.2.5)
•	•	•	KEY_POINTS	lexical element (see 7.29.2.6)
	•	•	LABEL	lexical element (see 7.30.3)
	•	•	LINE_COLOR	lexical element (see 7.29.2.7)
	•	•	LINE_TYPE	lexical element (see 7.29.2.8)
	•	•	REFRESH_ACTIONS	lexical element (see 7.29.2.9)
	•	•	Y_AXIS	lexical element (see 7.29.2.10)

7.16 LIKE**Purpose**

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

Lexical structure

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

The attributes are specified in Table 104.

Table 104 – LIKE attributes

Usage	Attribute	Description
m	identifier	is the name of the new EDD instance
m	reference	is a reference to an EDD instance which shall be copied
o	attribute-redefinition	specifies whether an attribute is added, deleted or redefined (see 7.15.2). Only existing attributes can be deleted or redefined
o	item-type	can be one of the elements listed in Table 42. Even if item-type is present it is not interpreted

7.17 LIST

7.17.1 General structure

Purpose

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

Lexical structure

LIST identifier [TYPE, CAPACITY, COUNT, HELP, LABEL]

The attributes of LIST are specified in Table 105.

Table 105 – LIST attributes

Usage	Attribute	Description
m	TYPE	lexical element (see 7.17.2.1)
o	CAPACITY	lexical element (see 7.17.2.2)
o	COUNT	lexical element (see 7.17.2.3)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)

7.17.2 Specific attributes

7.17.2.1 TYPE

Purpose

The TYPE attribute is a reference to a 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 106.

Table 106 – TYPE attribute

Usage	Attribute	Description
m	reference	is a reference to a VARIABLE instance, a RECORD instance, a VALUE_ARRAY instance, a COLLECTION instance or a REFERENCE_ARRAY instance

7.17.2.2 CAPACITY

Purpose

The CAPACITY attribute specifies the maximum number of elements that may be stored in the list. When a LIST is part of a FILE, the CAPACITY shall not be specified since the capacity is only limited by the resources of the EDD application.

Lexical structure

CAPACITY (unsigned_integer)<exp>

The attribute of CAPACITY is specified in Table 107.

Table 107 – CAPACITY, COUNT attribute

Usage	Attribute	Description
m	unsigned_integer	is an unsigned integer constant

7.17.2.3 COUNT

Purpose

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

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

Lexical structure

COUNT (unsigned_integer)<exp>

The attribute of COUNT is specified in Table 107.

7.18 MENU

7.18.1 General structure

Purpose

MENU organizes variables, methods, and other items specified in the EDDL into a hierarchical structure. A user application may use the MENU ITEMS to display and enter information in an organized and consistent fashion. The last elements of a hierarchical MENU structure shall be methods, variables or basic constructs referencing groups of them (e.g. COLLECTION OF VARIABLES).

If a submenu is displayed, the menu in which this sub menu is referenced shall be valid.

Lexical structure

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

The attributes of MENU are specified in Table 108.

Table 108 – MENU attribute

Usage	Attribute	Description
m	ITEMS	lexical element (see 7.18.2.1)
m	LABEL	lexical element (see 7.30.3)
o	ACCESS	lexical element (see 7.18.2.2)
o	ENTRY	lexical element (see 7.18.2.3)
o	HELP	lexical element (see 7.30.2)
o	POST_EDIT_ACTIONS	lexical element (see 7.18.2.4)
o	POST_READ_ACTIONS	lexical element (see 7.18.2.5)
o	POST_WRITE_ACTIONS	lexical element (see 7.18.2.6)
o	PRE_EDIT_ACTIONS	lexical element (see 7.18.2.7)
o	PRE_READ_ACTIONS	lexical element (see 7.18.2.8)
o	PRE_WRITE_ACTIONS	lexical element (see 7.18.2.9)
c	PURPOSE	lexical element (see 7.18.2.10). The PURPOSE shall be specified if the ENTRY attribute is used
o	ROLE	lexical element (see 7.18.2.11)
o	STYLE	lexical element (see 7.18.2.12)
o	VALIDITY	lexical element (see 7.18.2.13)

7.18.2 Specific attributes

7.18.2.1 ITEMS

Purpose

The ITEMS attribute specifies selected elements (VARIABLE, BLOCK_A PARAMETERS, elements of BLOCK_A PARAMETERS, COLLECTION, EDIT_DISPLAY, METHOD, RECORDS, REFERENCE_ARRAY, VALUE_ARRAY, VARIABLE_LIST, GRAPH, CHART and other MENUs) of the EDD which shall be displayed to the user plus optional qualifiers for each element. These qualifiers (DISPLAY_VALUE, HIDDEN, READ_ONLY, NO_LABEL, NO_UNIT, REVIEW) are used 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 [(reference, DISPLAY_VALUE, HIDDEN, READ_ONLY, NO_LABEL, NO_UNIT, REVIEW)<exp>, SEPARATOR, ROWBREAK, COLUMNBREAK, string]+

The attributes of ITEMS are specified in Table 109.

Table 109 – ITEMS attribute

Usage	Attribute	Description
m	reference	is a reference to a VARIABLE, a BLOCK_A PARAMETERS, an element of BLOCK_A PARAMETERS, a CHART, a COLLECTION, an EDIT_DISPLAY, a GRAPH, a GRID, an IMAGE instance, a MENU instance, a METHOD, RECORDS, a REFERENCE_ARRAY, a VALUE_ARRAY, or a VARIABLE_LIST
c/o	DISPLAY_VALUE	c only used with VARIABLE; o used in all other cases. Specifies that the value of the corresponding VARIABLE is displayed in addition to its LABEL
o	HIDDEN	specifies that the item is not displayed
c/o	READ_ONLY	c only used with VARIABLE; o used in all other cases. Specifies that the value shall be displayed but it may not be modified (regardless of its HANDLING attribute)
c/o	NO_LABEL	c only used with VARIABLE; o used in all other cases. Specifies that the label of the VARIABLE is not be displayed; DISPLAY_VALUE is implied
c/o	NO_UNIT	c only used with VARIABLE; o used in all other cases. Specifies that the engineering unit of the VARIABLE is not be displayed; DISPLAY_VALUE is implied
c/o	REVIEW	c only used with MENU; o used in all other cases. REVIEW is used to present a large quantity of data e.g. for review purpose
c/o	INLINE	c only used with IMAGE; o used in all other cases. 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 width of the MENU; otherwise, the IMAGE should be displayed inline with other ITEMS of the MENU.
o	SEPARATOR	specifies that the menu items before and after the SEPARATOR should be separated, e.g., via a horizontal line
c/o	COLUMNBREAK	c only used within a MENU whose STYLE attribute is WINDOW, DIALOG, PAGE, or GROUP; specifies a new column should be started
c/o	ROWBREAK	c only used within a MENU whose STYLE attribute is WINDOW, DIALOG, PAGE, or GROUP; specifies a new row should be started
o	string	specifies static text that should be displayed on the menu

7.18.2.2 ACCESS**Purpose**

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

Lexical structure

ACCESS [(OFFLINE, ONLINE) <exp>]

The attributes of ACCESS are specified in Table 110.

Table 110 – ACCESS attribute

Usage	Attribute	Description
s	OFFLINE	specifies that the values of the items displayed in the MENU are read locally
s	ONLINE	specifies that 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 should take place (typically after a single entry or MENU completion for an ONLINE ACCESS attribute, or via a separate menu to update the device).

7.18.2.3 ENTRY

Purpose

The ENTRY attribute specifies if a MENU represents an entry point into a hierarchical structure. If a MENU is defined as an entry point, this MENU shall also contain the PURPOSE attribute.

Lexical structure

ENTRY (ROOT) <exp>

The attribute of ENTRY is specified in Table 111.

Table 111 – ENTRY attribute

Usage	Attribute	Description
m	ROOT	specifies that the MENU may be used as an entry point

7.18.2.4 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 [(reference)<exp>]+, DEFINITION

The attributes of POST_EDIT_ACTIONS are specified in Table 112.

Table 112 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS attributes

Usage	Attribute	Description
o	reference	is a reference to a METHOD instance
o	DEFINITION	lexical element (see 7.30.1)

7.18.2.5 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 [(reference)<exp>]+, DEFINITION

The attributes of POST_READ_ACTIONS are specified in Table 112.

7.18.2.6 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 [(reference)<exp>]+, DEFINITION

The attributes of POST_WRITE_ACTIONS are specified in Table 112.

7.18.2.7 PRE_EDIT_ACTIONS**Purpose**

The PRE_EDIT_ACTIONS attribute specifies METHODS that shall be executed immediately 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 [(reference)<exp>]+, DEFINITION

The attributes of PRE_EDIT_ACTIONS are specified in Table 112.

7.18.2.8 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 [(reference)<exp>]+, DEFINITION

The attributes of PRE_READ_ACTIONS are specified in Table 112.

7.18.2.9 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 [(reference)<exp>]+, DEFINITION

The attributes of PRE_WRITE_ACTIONS are specified in Table 112.

7.18.2.10 PURPOSE

Purpose

The PURPOSE attribute defines the application purpose of the MENU. This indicates how the MENU is used, for example, for a tabular presentation, diagnostic actions, and upload/download list.

Lexical structure

PURPOSE [(CATALOG, DIAGNOSE, LOAD_TO_APPLICATION, LOAD_TO_DEVICE, MENU, PROCESS_VALUE, SIMULATION, string, TABLE)<exp>]+

The attributes of PURPOSE are specified in Table 113.

Table 113 – PURPOSE attribute

Usage	Attribute	Description
s	CATALOG	specifies that the elements of the ITEMS attribute of a MENU instance are references to catalog specific information (for example, an EDD variable containing the ordering number)
s	DIAGNOSE	specifies that the MENU is used for diagnostic actions
s	LOAD_TO_APPLICATION	specifies that the MENU is used for listing VARIABLES whose values are sent from the device to an application
s	LOAD_TO_DEVICE	specifies that the MENU is used for listing VARIABLES whose values are sent to the device
s	MENU	specifies a MENU which contains other MENU items or control items normally presented using horizontal rows
s	PROCESS_VALUE	specifies that the MENU is used to display process values
s	SIMULATION	specifies that the MENU is used for simulations actions
s	string	specifies a user-defined PURPOSE for the MENU
s	TABLE	specifies a MENU which contains other MENU items or VARIABLES normally presented using vertical columns

NOTE The PURPOSE attribute MENU should be used for a horizontal data and function representation (for example, function keys, application menus). TABLE should be used for a vertical data and function representation (for example, tables).

7.18.2.11 ROLE

Purpose

The ROLE attribute specifies in which role context the MENU is used. If a MENU contains sub-menus, only these submenus are displayed which have the same ROLE. If a MENU has no ROLE attribute, the MENU shall be displayed wherever it is referenced.

Lexical structure

ROLE [(string)<exp>]+

The attribute of ROLE is specified in Table 114.

Table 114 – ROLE attribute

Usage	Attribute	Description
m	string	specifies a string which indicates the ROLE of the MENU

NOTE An EDD may contain one or more of the following ROLES: MAINTENANCE, SPECIALIST, DIAGNOSE, HANDHELD, SERVICE, MANUFACTURER, OBSERVER.

7.18.2.12 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 7.18.2.1) is implied on each item on the MENU.

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

Lexical structure

STYLE [(DIALOG, WINDOW, PAGE, GROUP, MENU, TABLE, string)<exp>]

The attributes of STYLE are specified in Table 115.

Table 115 – STYLE attribute

Usage	Attribute	Description
s	DIALOG	specifies that no other MENU can be activated while this MENU 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
s	WINDOW	specifies that 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
s	PAGE	specifies that the items of the MENU should be displayed as a tabbed property page
s	GROUP	specifies that the items of the MENU should be displayed as a group box
s	MENU	specifies that the items of the MENU should be displayed as either a pulldown or popup menu
s	TABLE	specifies that the items of the MENU should be displayed in vertical columns
s	string	designates a tool specific MENU style. If a tool cannot interpret the string, this STYLE attribute shall be ignored

7.18.2.13 VALIDITY

Purpose

The VALIDITY attribute specifies whether a MENU is displayed or not. If the VALIDITY attribute is not defined the default value is TRUE.

NOTE VALIDITY is normally expressed using a conditional expression (see 7.32).

Lexical structure

VALIDITY [(FALSE, TRUE)<exp>]

The attributes of VALIDITY are specified in Table 116.

Table 116 – VALIDITY attributes

Usage	Attribute	Description
s	FALSE	specifies that the MENU is not displayed
s	TRUE	specifies that the MENU is displayed

7.18.3 Sequence diagrams for actions

Figure 28 to Figure 34 show the sequence diagrams for actions related to MENUS and their contained VARIABLES and METHODS. Actions are executed and completed in sequence moving down a diagram. If any action fails or aborts for an unplanned reason, the remaining actions are not executed and the MENU is cancelled.

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

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

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

In the case of online access, VARIABLE values may be changed and immediately written unless they are in a WRITE_AS_ONE relation. Any WRITE_AS_ONE entities are written together. After the WRITE_AS_ONE VARIABLES have been written, the POST_WRITE_ACTIONS are executed.

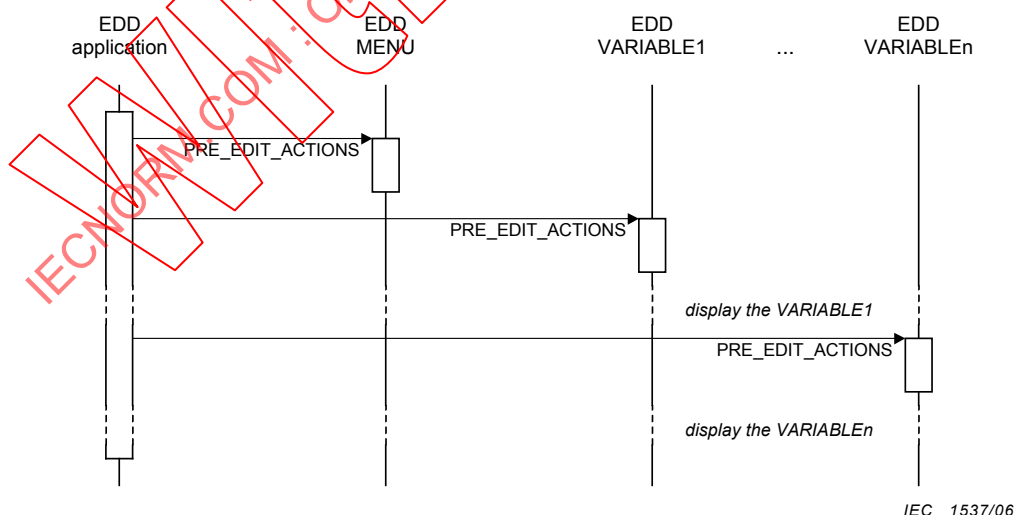


Figure 28 – MENU activation (ACCESS OFFLINE)

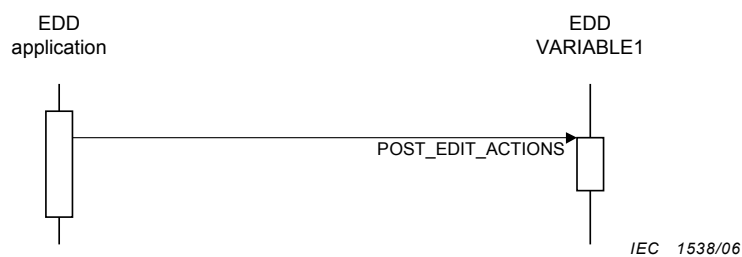


Figure 29 – Action performed after a new value is entered

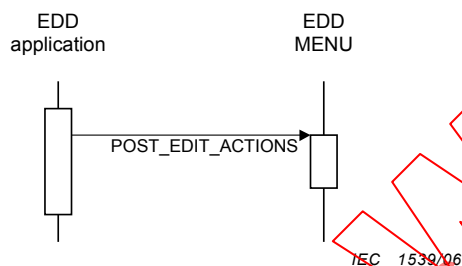
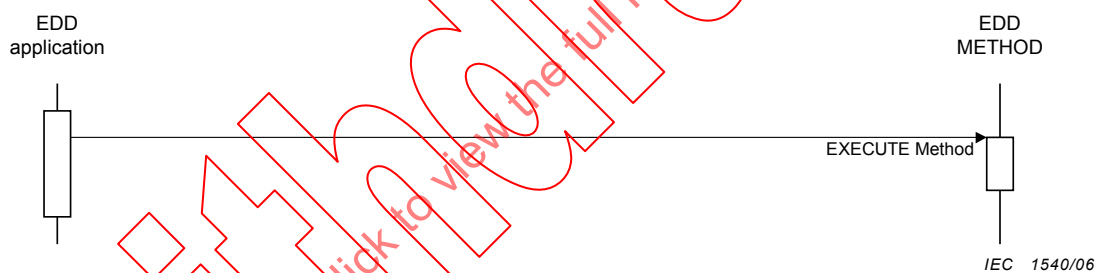
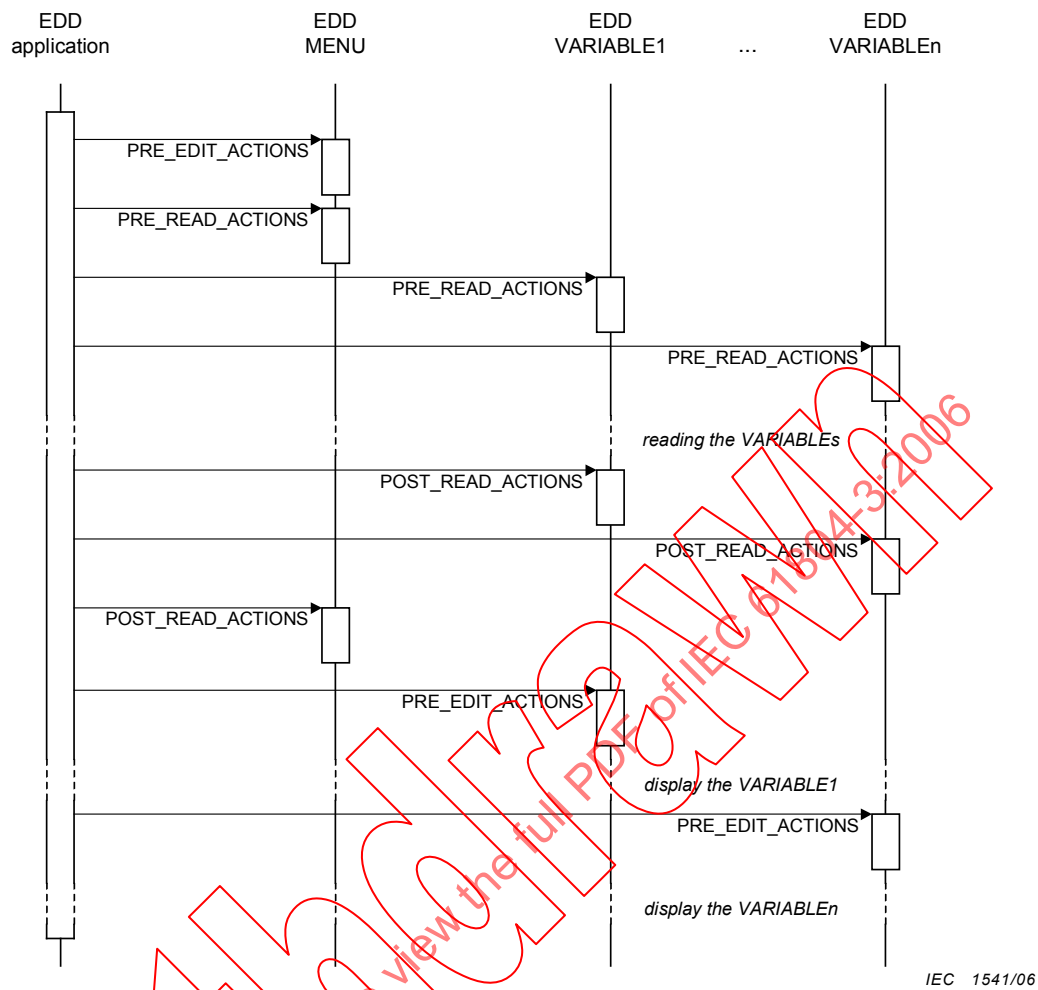


Figure 30 – Action performed after all VARIABLE inputs of the MENU are accepted (ACCESS OFFLINE)



In case of online access, changed VARIABLE values shall be written to the device before executing the METHOD. After execution of the METHOD the values are read back from the device and displayed.

Figure 31 – Method execution



IEC 1541/06

- a For the VARIABLES that are read online from a device the EDD application executes appropriate COMMANDs. VARIABLES may include pre-read/write actions that shall precede the appropriate COMMAND. VARIABLES post-read/write actions shall be executed following the appropriate COMMAND.
- b Each VARIABLE or other entity is displayed to the user after its PRE_EDIT_ACTIONS are complete.
- c PRE_READ_ACTIONS and POST_READ_ACTIONS for different VARIABLES may be in any convenient order; they shall all be complete before the MENU POST_READ_ACTIONS are executed.

Figure 32 – MENU activation (ACCESS ONLINE)

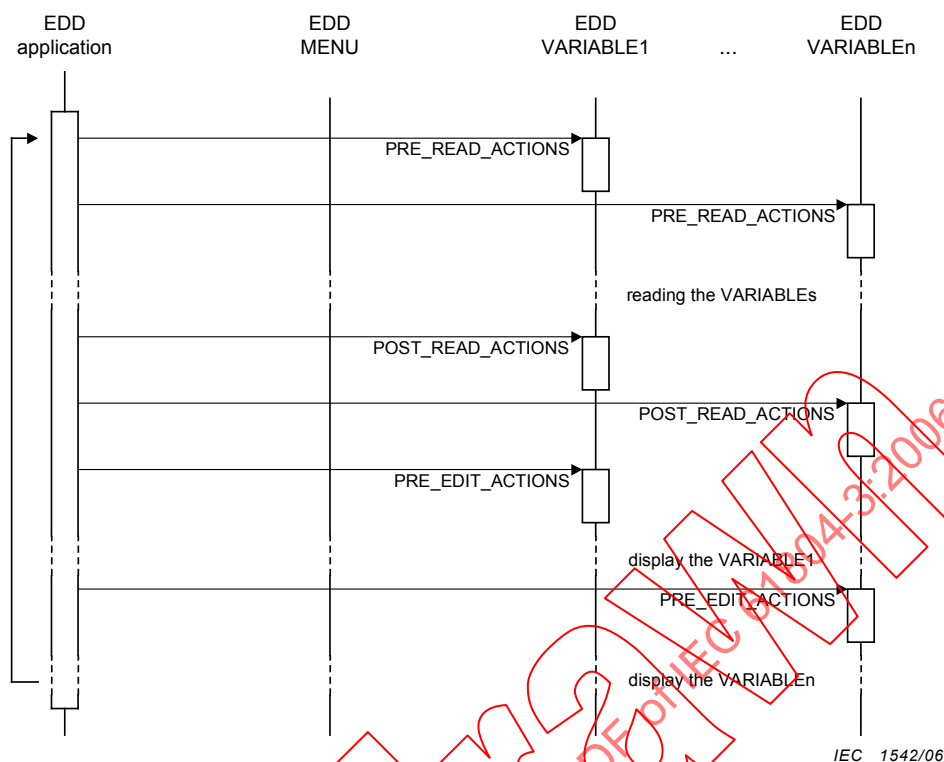
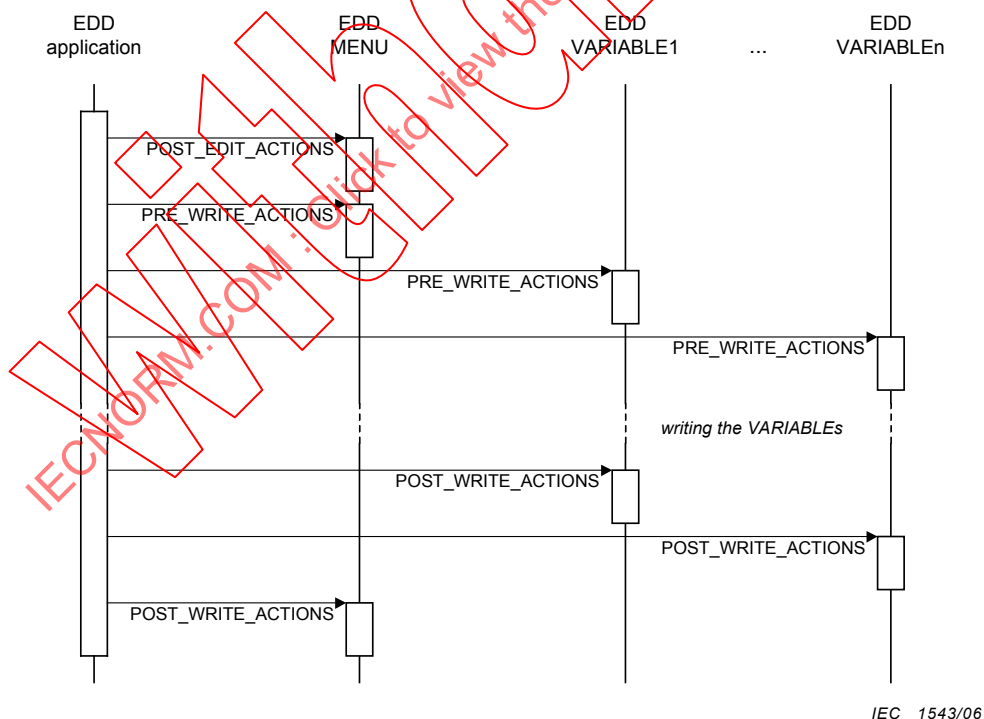


Figure 33 – Cyclic reading of dynamic VARIABLES (ACCESS ONLINE)



METHODS are executed after all VARIABLE inputs are accepted. Only the VARIABLE values are written to the device which are changed. After the changes are written, the VARIABLES shall be read for consistency purposes.

Figure 34 – Action performed after all VARIABLE inputs of the MENU are accepted (ACCESS ONLINE)

7.19 METHOD

7.19.1 General structure

Purpose

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

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

Lexical structure

METHOD identifier [DEFINITION, LABEL, ACCESS, CLASS, HELP, TYPE, VALIDITY]

The attributes of METHOD are specified in Table 117.

Table 117 – METHOD attributes

Usage	Attribute	Description
m	DEFINITION	lexical element (see 7.30.1)
m	LABEL	lexical element (see 7.30.3)
o	ACCESS	lexical element (see 7.19.2.1)
o	CLASS	lexical element (see 7.19.2.2)
o	HELP	lexical element (see 7.30.2)
o	TYPE	lexical element (see 7.19.2.3)
o	VALIDITY	lexical element (see 7.19.2.3)

7.19.2 Specific attributes

7.19.2.1 ACCESS

Purpose

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

Lexical structure

ACCESS (OFFLINE, ONLINE) <exp>

The attributes of ACCESS are specified in Table 118.

Table 118 – ACCESS attributes

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

7.19.2.2 CLASS

Purpose

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

Lexical structure

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

The attributes of CLASS are specified in Table 139.

7.19.2.3 TYPE

Purpose

The TYPE attribute specifies the type of the returned value.

Lexical structure

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

The attributes of VALIDITY are specified in Table 120.

Table 119 – TYPE attributes

Usage	Attribute	Description
s	char	specifies that the METHOD returns a value of type char
s	unsigned char	specifies that the METHOD returns a value of type unsigned char
s	int	specifies that the METHOD returns a value of type int
s	unsigned int	specifies that the METHOD returns a value of type unsigned int
s	long	specifies that the METHOD returns a value of type long
s	unsigned long	specifies that the METHOD returns a value of type unsigned long
s	float	specifies that the METHOD returns a value of type float
s	double	specifies that the METHOD returns a value of type double
s	DD_STRING	specifies that the METHOD returns a value of type DD_STRING

7.19.2.4 VALIDITY

Purpose

The VALIDITY attribute specifies whether the METHOD is executed or not. A METHOD without VALIDITY attribute is always valid.

NOTE VALIDITY is normally expressed using a conditional expression (see 7.32).

Lexical structure

VALIDITY [(FALSE, TRUE)<exp>]

The attributes of VALIDITY are specified in Table 120.

Table 120 – VALIDITY attributes

Usage	Attribute	Description
s	FALSE	specifies that the METHOD shall not be executed
s	TRUE	specifies that the METHOD shall be executed

7.20 PROGRAM

7.20.1 General structure

Purpose

PROGRAM allows remote control for the execution of a program.

EXAMPLE A device could download a PROGRAM into a domain of another device using a download service and then remotely operate the PROGRAM by issuing a service request. PROGRAMs should be created, deleted, started, stopped, etc.

Lexical structure

PROGRAM identifier [ARGUMENT, RESPONSE_CODES]

The attributes of PROGRAM are specified in Table 121.

Table 121 – PROGRAM attributes

Usage	Attribute	Description
o	ARGUMENT	lexical element (see 7.20.2.1)
o	RESPONSE_CODES	lexical element (see 7.30.5)

7.20.2 Specific attributes

7.20.2.1 ARGUMENT

Purpose

ARGUMENTs can be sent to the program during start and resume operations. The PROGRAM arguments are described by the ARGUMENT attribute. An octet string containing the values of all arguments will be sent to the program invocation object when it is started or resumed by the EDD application.

Lexical structure

ARGUMENT [(data-item)<exp>] +

The attribute of ARGUMENT is specified in Table 122.

Table 122 – ARGUMENT attribute

Usage	Attribute	Description
m	data-item	is either an unsigned integer constant or a VARIABLE. If a data item is an unsigned integer constant, the value of the constant appears at that position in the data field. Constant data items occupy two octets of the data field and therefore shall be in the range 0 through 65535 inclusive. If a data item is a VARIABLE, the value of the variable appears at that position in the data field. If the data field of the PROGRAM service is empty, the arguments can be omitted

7.20.2.2 General

The PROGRAM has only one specific attribute.

7.21 RECORD

Purpose

A RECORD is a logical group of VARIABLES. Each item in the RECORD is a reference to a VARIABLE and may be of different data types.

NOTE A RECORD may be transferred as one communication object.

Lexical structure

RECORD identifier [MEMBERS, LABEL, HELP, RESPONSE_CODES]

The attributes of RECORD are specified in Table 123.

Table 123 – RECORD attributes

Usage	Attribute	Description
m	MEMBERS	lexical element (see 7.30.4)
m	LABEL	lexical element (see 7.30.3)
o	HELP	lexical element (see 7.30.2)
o	RESPONSE_CODES	lexical element (see 7.30.5)

7.22 REFERENCE_ARRAY

7.22.1 General structure

Purpose

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

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

Lexical structure

REFERENCE_ARRAY identifier [item-type, ELEMENTS, HELP, LABEL]

The attributes of REFERENCE_ARRAY are specified in Table 124.

Table 124 – REFERENCE_ARRAY attribute

Usage	Attribute	Description
m	item-type	specifies the type of items listed within the ELEMENTS attribute. It shall be one of the basic elements listed in Table 42
m	ELEMENTS	lexical element (see 7.22.2.1)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)

7.22.2 Specific attributes

7.22.2.1 ELEMENTS

Purpose

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

Lexical structure

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

The attributes of ELEMENTS are specified in Table 125.

Table 125 – ELEMENTS attribute

Usage	Attribute	Description
m	integer	specifies the number by which the item may be referenced. Each number shall be uniquely defined within an REFERENCE_ARRAY construct
m	reference	is a reference to a data-item instance
o	description	provides a short description for the item
o	help	specifies help text for the item

7.22.2.2 General

The REFERENCE_ARRAY has only one specific attribute.

7.23 Relations

7.23.1 General structure

Purpose

Relations specify relationships between instances of VARIABLES. The following types of relations are defined:

- REFRESH (see 7.23.2)
- UNIT (see 7.23.3)
- WRITE_AS_ONE (see 7.23.4)

7.23.2 REFRESH

Purpose

A REFRESH relation is used during online access and specifies a set of VARIABLES which should be refreshed (read from the device) whenever a VARIABLE from a specified set is modified.

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

Lexical structure

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

The attributes of REFRESH are specified in Table 126.

Table 126 – REFRESH attributes

Usage	Attribute	Description
m	cause-parameter	specifies a list of VARIABLE instances
m	effected-parameter	specifies a list of VARIABLE instances, which shall be read, if one of the cause parameters has been modified. A VARIABLE of the effected parameter list may also be in the cause parameter list of another REFRESH relation

7.23.3 UNIT

Purpose

A UNIT relation specifies a reference to a VARIABLE holding a unit code and a list of dependent VARIABLES. When the VARIABLE holding the unit code is modified, the list of effected VARIABLES/AXISs using that unit code shall be refreshed. When an effected VARIABLE/AXIS is displayed, the value of its unit code shall also be displayed.

Lexical structure

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

The attributes of UNIT are specified in Table 127.

Table 127 – UNIT attributes

Usage	Attribute	Description
m	cause-parameter	is a reference to a VARIABLE instance which contains the unit code
m	effected-parameter	specifies a list of VARIABLE/AXIS instances using that unit code

7.23.4 WRITE_AS_ONE

Purpose

A WRITE_AS_ONE relation specifies a list of VARIABLES that shall be modified as a group by the EDD application. This relation is used when a device requires specific VARIABLES to be examined and modified at the same time for proper operation.

NOTE This relation does not necessarily mean the VARIABLES are written to the device at the same time. Not all VARIABLES sent to the device at the same time are necessarily part of a WRITE_AS_ONE relation. It is up to the EDD application to determine how the updates to the parameters of this relation are passed to the device.

Lexical structure

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

The attribute of WRITE_AS_ONE is specified in Table 128.

Table 128 – WRITE_AS_ONE attribute

Usage	Attribute	Description
m	reference	specifies a list of VARIABLES, VALUE_ARRAYs and RECORDs instances that shall be modified in a group with other members of the WRITE_AS_ONE relation

7.24 RESPONSE_CODES

Purpose

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

Lexical structure

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

The attributes of RESPONSE_CODES are specified in Table 129.

Table 129 – RESPONSE_CODES attributes

Usage	Attribute	Description
m	description	specifies a string that will be displayed when the RESPONSE_CODE is returned by the device
m	integer	specifies the returned value of the device to identify the RESPONSE_CODE
s	DATA ENTRY ERROR	the application layer service was rejected because the data sent was invalid
s	DATA ENTRY WARNING	the application layer service was accepted and processed with a slightly modified version of the data sent
s	MISC ERROR	the application layer service was rejected
s	MISC WARNING	the application layer service was accepted and processed as specified and there is additional information, unrelated to the application layer service, in which the user might be interested
s	MODE ERROR	the application layer service was rejected because the field device was in a mode in which the application layer service could not be executed
s	PROCESS ERROR	the application layer service was rejected because a process applied to the field device was invalid
s	SUCCESS	the application layer service was accepted and processed as specified
o	help	specifies help text for the item

7.25 SOURCE

7.25.1 General structure

Purpose

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

Lexical structure

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

The attributes of SOURCE are specified in Table 130.

Table 130 – SOURCE attributes

Usage	Attribute	Description
m	MEMBERS	lexical element (see 7.30.4)
o	EMPHASIS	lexical element (see 7.25.2.1)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)
o	LINE_COLOR	lexical element (see 7.25.2.2)
o	LINE_TYPE	lexical element (see 7.25.2.3)
o	Y_AXIS	lexical element (see 7.25.2.4)
o	EXIT_ACTIONS	lexical element (see 7.25.2.5)
o	INIT_ACTIONS	lexical element (see 7.25.2.6)
o	REFRESH_ACTIONS	lexical element (see 7.25.2.7)

7.25.2 Specific attributes

7.25.2.1 EMPHASIS

Purpose

The EMPHASIS attribute specifies whether or not the SOURCE should be emphasized when it is displayed. The way in which a SOURCE is emphasized is defined by the EDD application. By default, a SOURCE without an EMPHASIS attribute should not be emphasized.

Lexical structure

EMPHASIS (FALSE, TRUE) <exp>

The attributes of EMPHASIS are specified in Table 131.

Table 131 – EMPHASIS attribute

Usage	Attribute	Description
s	FALSE	specifies that the SOURCE should not be emphasized
s	TRUE	specifiers that the SOURCE should be emphasized

7.25.2.2 LINE_COLOR

Purpose

The LINE_COLOR attribute specifies the color that should be used to display the SOURCE. By default, a SOURCE without a LINE_COLOR attribute will be displayed using a default color defined by the EDD application.

Lexical structure

LINE_COLOR (integer, reference) <exp>

Colors are specified via a RGB value, as defined in Cascading Style Sheets, Level 2.

The attributes of LINE_COLOR are specified in Table 132.

Table 132 – LINE_COLOR attribute

Usage	Attribute	Description
s	integer	is the color to be used to display the SOURCE
s	reference	is a reference to a VARIABLE instance containing the color to be used to display the SOURCE

7.25.2.3 LINE_TYPE

Purpose

The LINE_TYPE attribute specifies the type of data contained in the SOURCE. Every SOURCE with the same LINE_TYPE attribute should be displayed in the same fasion (i.e., line color, line pattern, line thickness, etc). By default, a SOURCE without a LINE_TYPE attribute uses the DATA line style.

Lexical structure

LINE_TYPE ([DATA, n]*, LOW_LOW_LIMIT, LOW_LIMIT, HIGH_LIMIT, HIGH_HIGH_LIMIT) <exp>

The attributes of LINE_TYPE are specified in Table 133.

Table 133 – LINE_TYPE attribute

Usage	Attribute	Description
s	DATA	specifies the SOURCE contains data
c	n	is an increasing integer starting with one, which identifies distinct DATA line types if more than one is present
s	LOW_LOW_LIMIT	specifies the SOURCE contains a low low limit
s	LOW_LIMIT	specifies the SOURCE contains a low limit
s	HIGH_LIMIT	specifies the SOURCE contains a high limit
s	HIGH_HIGH_LIMIT	specifies the SOURCE contains a high high limit

7.25.2.4 Y_AXIS

Purpose

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

Lexical structure

Y_AXIS (reference) <exp>

The attribute of Y_AXIS is specified in Table 134.

Table 134 – Y_AXIS attribute

Usage	Attribute	Description
m	reference	is a reference to an AXIS instance

7.25.2.5 EXIT_ACTIONS

Purpose

EXIT_ACTIONS 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 [(reference) <exp>]+, DEFINITION

The attributes of EXIT_ACTIONS are specified in Table 167.

7.25.2.6 INIT_ACTIONS

Purpose

INIT_ACTIONS are executed before the SOURCE value is initially displayed by a CHART.

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

Lexical structure

INIT_ACTIONS [(reference) <exp>]+, DEFINITION

The attributes of INIT_ACTIONS are specified in Table 167.

7.25.2.7 REFRESH_ACTIONS

REFRESH_ACTIONS are executed prior to the display of the value on the CHART in which the SOURCE is being displayed.

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

Lexical structure

REFRESH_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of REFRESH_ACTIONS are specified in Table 167.

7.26 VALUE_ARRAY

7.26.1 General structure

Purpose

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

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

Lexical structure

VALUE_ARRAY identifier [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, RESPONSE_CODES]

The attributes of VALUE_ARRAY are specified in Table 135.

Table 135 – VALUE_ARRAY attributes

Usage	Attribute	Description
m	LABEL	lexical element (see 7.30.3)
m	NUMBER_OF_ELEMENTS	lexical element (see 7.26.2.1)
m	TYPE	lexical element (see 7.26.2.2)
o	HELP	lexical element (see 7.30.2)
o	RESPONSE_CODES	lexical element (see 7.30.5)

7.26.2 Specific attributes

7.26.2.1 NUMBER_OF_ELEMENTS

Purpose

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

Lexical structure

NUMBER_OF_ELEMENTS integer, expression)

The attribute of NUMBER_OF_ELEMENTS is specified in Table 136.

Table 136 – NUMBER_OF_ELEMENTS attribute

Usage	Attribute	Description
s	integer	specifies the number of elements of a VALUE_ARRAY
s	expression	is an expression which shall be evaluated to a non-negative integer that is within the index range of the NUMBER_OF_ELEMENTS attribute (for the description of expressions, see 7.34.7).

7.26.2.2 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 VALUE_ARRAY elements.

Lexical structure

TYPE reference

The TYPE attribute is specified in Table 137.

Table 137 – TYPE attribute

Usage	Attribute	Description
m	reference	is a reference to an EDD basic construct instance

7.27 VARIABLE**7.27.1 General structure****Purpose**

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

Lexical structure

VARIABLE identifier, [CLASS, LABEL, TYPE, HELP, CONSTANT_UNIT, DEFAULT_VALUE, PRE_EDIT_ACTIONS, POST_EDIT_ACTIONS, PRE_READ_ACTIONS, POST_READ_ACTIONS, PRE_WRITE_ACTIONS, POST_WRITE_ACTIONS, REFRESH_ACTIONS, READ_TIMEOUT, WRITE_TIMEOUT, RESPONSE_CODE, HANDLING, VALIDITY]

The attributes of VARIABLE are specified in Table 138.

Table 138 – VARIABLE attributes

Usage	Attribute	Description
m	CLASS	lexical element (see 7.27.2.1)
m	LABEL	lexical element (see 7.30.3)
m	TYPE	lexical element (see 7.27.2.2)
o	CONSTANT_UNIT	lexical element (see 7.27.2.3)
o	DEFAULT_VALUE	lexical element (see X.X.X)
o	HANDLING	lexical element (see 7.27.2.4)
o	HELP	lexical element (see 7.30.2)
o	INITIAL_VALUE	lexical element (see X.X.X)
o	POST_EDIT_ACTIONS	lexical element (see 7.27.2.6)

Usage	Attribute	Description
o	POST_READ_ACTIONS	lexical element (see 7.27.2.8)
o	POST_WRITE_ACTIONS	lexical element (see 7.27.2.9)
o	PRE_EDIT_ACTIONS	lexical element (see 7.27.2.10)
o	PRE_READ_ACTIONS	lexical element (see 7.27.2.11)
o	PRE_WRITE_ACTIONS	lexical element (see 7.27.2.12)
o	REFRESH_ACTIONS	lexical element (see 7.27.2.13)
o	READ_TIMEOUT	lexical element (see 7.27.2.14)
o	RESPONSE_CODE	lexical element (see 7.30.5)
o	STYLE	lexical element (see 7.27.2.15)
o	VALIDITY	lexical element (see 7.27.2.16)
o	WRITE_TIMEOUT	lexical element (see 7.27.2.13)

7.27.2 Specific attributes

7.27.2.1 CLASS

Purpose

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

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

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

Lexical structure

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

The attributes of CLASS are specified in Table 139.

Table 139 – CLASS attributes

Usage	Attribute	Description
s	ALARM	specifies that VARIABLE contains alarm limits
s	ANALOG_INPUT	specifies that VARIABLE is part of an analog input block
s	ANALOG_OUTPUT	specifies that VARIABLE is part of an analog output block
s	COMPUTATION	specifies that VARIABLE is part of a computation block
s	CONTAINED	specifies that VARIABLE represents the physical characteristics of the device
s	CORRECTION	specifies that VARIABLE is part of the correction block
s	DEVICE	specifies that VARIABLE represents the physical characteristics of the device
s	DIAGNOSTIC	specifies that VARIABLE indicates the device status
s	DIGITAL_INPUT	specifies that VARIABLE is part of a digital input block
s	DIGITAL_OUTPUT	specifies that VARIABLE is part of a digital output block
s	DISCRETE_INPUT	specifies that VARIABLE is part of a discrete input block

Usage	Attribute	Description
s	DISCRETE_OUTPUT	specifies that VARIABLE is part of a discrete output block
s	DYNAMIC	specifies that VARIABLE is modified by the device without stimulus from the network
s	FREQUENCY_INPUT	specifies that VARIABLE is part of a frequency input block
s	FREQUENCY_OUTPUT	specifies that VARIABLE is part of a frequency output block
s	HART	specifies that VARIABLE is part of the HART block which characterize the HART interface. There is only one HART block
s	INPUT	specifies that VARIABLE is part of an input block. An input block is a special kind of computation block which does unit conversions, scaling, and damping. The parameter of the input block parameters can be determined by the output of another block
s	LOCAL	specifies that VARIABLE is locally used by the EDD application. Local VARIABLES are not stored in a device, but they can be sent to a device
s	LOCAL_DISPLAY	specifies that VARIABLE is part of the local display block. A local display block contains the VARIABLES associated with the local interface (keyboard, display, etc.) of the device
s	OPERATE	specifies that VARIABLE is used to control a block's operation
s	OUTPUT	specifies that VARIABLE is part of the output block. The values of output VARIABLES may be accessed by another block input
s	SERVICE	specifies that VARIABLE is used when performing routine maintenance
s	TUNE	specifies that VARIABLE is used to tune the algorithm of a block

7.27.2.2 TYPE

7.27.2.2.1 General structure

Purpose

The TYPE attribute describes the format of the VARIABLES value. If TYPE is specified without DEFAULT_VALUE and INITIAL_VALUE, the column "Initial value" in Table 140 shows the initial values.

Lexical structure

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

The attributes of TYPE are specified in Table 140.

Table 140 – TYPE attributes

Usage	Attribute	Description	Initial value
s	DOUBLE	arithmetic type (see 7.27.2.2.2.1)	1
s	FLOAT	arithmetic type (see 7.27.2.2.2.1)	1
s	INTEGER	arithmetic type (see 7.27.2.2.2.1)	1
s	UNSIGNED_INTEGER	arithmetic type (see 7.27.2.2.2.1)	1
s	DATE	date/time types (see 7.27.2.2.3)	
s	DATE_AND_TIME	date/time types (see 7.27.2.2.3)	
s	DURATION	date/time types (see 7.27.2.2.3)	
s	TIME	date/time types (see 7.27.2.2.3)	00:00:00
s	TIME_VALUE	date/time types (see 7.27.2.2.3)	
s	BIT_ENUMERATED	enumeration type (see 7.27.2.2.3.1)	0
s	ENUMERATED	enumeration type (see 7.27.2.2.3.2)	the value of the first

Usage	Attribute	Description	Initial value
s	INDEX	index type (see 7.27.2.2.3.3)	
s	OBJECT_REFERENCE	object references (see 7.27.2.2.4)	
s	ASCII	string type (see 7.27.2.2.5)	
s	BITSTRING	string type (see 7.27.2.2.5)	
s	EUC	string type (see 7.27.2.2.5)	
s	OCTET	string type (see 7.27.2.2.5)	
s	PACKED_ASCII	string type (see 7.27.2.2.5)	
s	PASSWORD	string type (see 7.27.2.2.5)	
s	VISIBLE	string type (see 7.27.2.2.5)	

7.27.2.2.2 Specific attributes

7.27.2.2.2.1 DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Purpose

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

These types are known collectively as the arithmetic types.

Lexical structure — DOUBLE

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

Lexical structure — FLOAT

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

Lexical structure — INTEGER

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

Lexical structure — UNSIGNED_INTEGER

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

The attributes of the arithmetic types are specified in Table 141.

Table 141 – DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER attributes

Usage	Attribute	Description
o	DEFAULT_VALUE	Purpose The DEFAULT_VALUE specifies the default setting for the VARIABLE. The DEFAULT_VALUE is available for all EDDL types. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as an INITIAL_VALUE. The DEFAULT_VALUE can also be built by an expression (see 7.35). Lexical structure DEFAULT_VALUE (double)<exp> DEFAULT_VALUE (float)<exp> DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	description	is a short description of the item
o	DISPLAY_FORMAT	Purpose A DISPLAY_FORMAT specifies how the value of the VARIABLE is displayed. The DISPLAY_FORMAT strings are conversion specifiers of the ANSI C (see ISO/IEC 9899) functions printf and scanf, respectively Lexical structure DISPLAY_FORMAT (string)<exp>
o	EDIT_FORMAT	Purpose An EDIT_FORMAT specifies the format for the editing. The EDIT_FORMAT strings are conversion specifiers of the ANSI C (see ISO/IEC 9899) functions printf and scanf, respectively Lexical structure EDIT_FORMAT (string)<exp>
o	help	specifies help text for the item
o	INITIAL_VALUE	Purpose When a device is instantiated for the first time, the INITIAL_VALUE of a VARIABLE is displayed. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned Lexical structure INITIAL_VALUE (double) INITIAL_VALUE (float) INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)
o	MAX_VALUE	Purpose The MAX_VALUE specifies the upper bound of values to which a VARIABLE shall be set. If the VARIABLE is CLASS DYNAMIC, the device should set the value of the VARIABLE inside the range specified by its minimum and maximum values A VARIABLE may have more than one maximum value, for instance, to specify more than one range of validity. When there are multiple maximum values, an additional integer is used to number each MAX_VALUE Every MAX_VALUE shall have an equivalent MIN_VALUE. The range specified with a pair of MIN_VALUE and MAX_VALUE may not overlap another specified range. If a MIN_VALUE/MAX_VALUE has no equivalent MAX_VALUE/MIN_VALUE, the upper/lower bound is unlimited and need not be specified. The MAX_VALUE can also be built by an expression (see 7.35) Lexical structure MAX_VALUE (double, integer)<exp> MAX_VALUE (float, integer)<exp> MAX_VALUE (integer, integer)<exp> MAX_VALUE (unsigned_integer, integer)<exp> MAX_VALUE (expression)<exp>
c	m	is an increasing integer starting with one, which identifies the MAX_VALUE if more than one is present

Usage	Attribute	Description
o	MIN_VALUE	Purpose The MIN_VALUE specifies the lower bound of values to which a VARIABLE shall be set. If the VARIABLE is CLASS DYNAMIC, the device should set the value of the VARIABLE inside the range specified by its minimum and maximum values A VARIABLE may have more than one minimum value, for instance, to specify more than one range of validity. When there are multiple minimum values, an additional integer is used to number each MIN_VALUE Every MIN_VALUE shall have an equivalent MAX_VALUE. The range specified with a pair of MIN_VALUE and MAX_VALUE may not overlap another specified range. If a MIN_VALUE/MAX_VALUE has no equivalent MAX_VALUE/MIN_VALUE, the upper/lower bound is unlimited and need not be specified. The MIN_VALUE can also be built by an expression (see 7.35) Lexical structure MIN_VALUE (double, integer)<exp> MIN_VALUE (float, integer)<exp> MIN_VALUE (integer, integer)<exp> MIN_VALUE (unsigned_integer, integer)<exp> MIN_VALUE (expression)<exp>
c	n	is an increasing integer starting with one, which identifies the MIN_VALUE if more than one is present
o	SCALING_FACTOR	Purpose The SCALING_FACTOR indicates that the displayed value of the VARIABLE is not the value returned by a device. The displayed value is the value returned by a device multiplied by a factor. Therefore, the EDDL processor shall multiply the value of the VARIABLE returned by a device with its SCALING_FACTOR before it is displayed (or before it is used in any other way). The SCALING_FACTOR shall not equal zero. The SCALING_FACTOR can also be built by an expression (see 7.35) Lexical structure SCALING_FACTOR (double)<exp> SCALING_FACTOR (expression)<exp>
o	size	specifies the size of the data type in octets. Size is an integer constant greater than zero and has no upper bound. This value is optional. The default is 1
o	value	is a constant that specifies the value. The type depends on the data type specified with the TYPE attribute which is optional, but if defined a description attribute is required. Equal values are not allowed

7.27.2.2.3 DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

Purpose

VARIABLES of data type DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE are used to specify dates:

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

Lexical structure — DATE

DATE specifies that VARIABLE is an integer of three octets. The coding of the octets is specified in D.5.2.

DATE DEFAULT_VALUE, INITIAL_VALUE

Lexical structure – DATE_AND_TIME

DATE_AND_TIME specifies that VARIABLE is an integer of seven octets. The coding of the octets is specified in D.5.3.

DATE_AND_TIME DEFAULT_VALUE, INITIAL_VALUE

Lexical structure — DURATION

DURATION specifies that VARIABLE is an integer of 4-6 octets. The first four bit shall have the value zero. Bit 0 to 27 represents the time in ms. The optional number of days is a 16-bit binary value. The coding of the octets is specified in D.5.4.

DURATION DEFAULT_VALUE, INITIAL_VALUE

Lexical structure — TIME

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

The date is stated in days relatively to 1984-01-01. On the first day of January 1984, date counting starts with the value zero. The coding of the octets is specified in D.5.5.

TIME DEFAULT_VALUE, INITIAL_VALUE

Lexical structure – TIME_VALUE

TIME_VALUE specifies that VARIABLE is an integer of eight octets. It is a fixed point number, which is the time in multiples of 1/32 ms. The coding of the octets is specified in D.5.6.

TIME_VALUE DEFAULT_VALUE, INITIAL_VALUE

The attributes of the DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE types are specified in Table 141.

7.27.2.2.3.1 BIT_ENUMERATED

Purpose

Variables of data type BIT_ENUMERATED are unsigned integer constants which identifies a bit of the VARIABLE value with its position.

Lexical structure

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

The attributes of the BIT_ENUMERATED type are specified in Table 142.

Table 142 – BIT_ENUMERATED attributes

Usage	Attribute	Description
m	description	see Table 141
m	value	is an unsigned integer that specifies a bit position. If the value of the VARIABLE supplies at the specified bit positions a logical one, the associated description is displayed. Equal values are not allowed
o	actions	specifies actions that will be executed when the bit is set
o	function	specifies the functional class of the bit. The functional class of a bit is the same as the CLASS of a VARIABLE. If no function is specified, the value of the function class defaults to the class of the variable. Therefore, if all the bits have the same function, then the CLASS of the VARIABLE shall be specified

Usage	Attribute	Description
o	help	see Table 141
o	size	see Table 141
o	status-class	specifies the meaning of the bit. A status bit may belong to more than one status-class (see Table 143). If the VARIABLE is not a status octet, then the bits do not have status classes
o	DEFAULT_VALUE	Purpose The DEFAULT_VALUE specifies the default setting for the VARIABLE. The DEFAULT_VALUE is available for all EDDL types. If no INITIAL_VALUE is defined, the DEFAULT_VALUE is used as a INITIAL_VALUE. The DEFAULT_VALUE can also be built by an expression (see 7.35) Lexical structure DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	INITIAL_VALUE	Purpose When a device is instantiated for the first time, the INITIAL_VALUE of a VARIABLE is displayed. Often the DEFAULT_VALUE is conditioned and depends on other parameter settings. In this case the INITIAL_VALUE is used to reconstitute the default settings. The INITIAL_VALUE is available for all types and shall not be conditioned Lexical structure INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)

Table 143 shows the status-classes for a VARIABLE of type BIT_ENUMERATED.

Table 143 – status-class attributes

Usage	Attribute	Description
m	DATA	indicated an invalid data configuration
m	HARDWARE	indicates a hardware failure
m	MISC	indicates a miscellaneous condition
m	MODE	indicates that the device is in a particular mode
m	PROCESS	indicates a problem with the process connected to the field device
m	SOFTWARE	indicates a software failure
m	CORRECTABLE	indicates that the bit can be cleared by the EDD application or the connected process
m	SELF_CORRECTING	indicates that the bit will be reset without further intervention
m	UNCORRECTABLE	indicates that the bit is neither self-correcting nor correctable
m	EVENT	indicates a one-time event
m	STATE	indicates that the device is in a particular state.
m	COMM_ERROR	indicates a communications failure in the other device
o	IGNORE_IN_HANDHELD	indicates that a bit should be ignored in hand-held communicators
o	IGNORE_IN_TEMPORARY_MASTER	indicates that a bit should be ignored in temporary master devices
m	MORE	indicates that there is additional status information available from the device
o	ALL	indicates the impact on all the outputs (see Table 144)
o	AO	indicates the impact on one of the analog outputs (see Table 144)
m	BAD	indicates that the output is unreliable and should not be used for control (see Table 144)
o	DV	indicates an impact on one of the dynamic variables (see Table 144)
o	TV	indicates the impact on one of the transmitter variables (see Table 144)
m	DETAIL	indicates that the bit is summarized elsewhere in a bit of class summary
m	SUMMARY	indicates that the bit is a logical combination of other bits of class detail

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

Lexical structure — ALL

ALL [AUTO_BAD, AUTO_GOOD, MANUAL_BAD, MANUAL_GOOD]

Lexical structure — AO

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

Lexical structure — DV

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

Lexical structure — TV

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

The attributes are specified in Table 144.

Table 144 – ALL, AO, DV, TV attributes

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

7.27.2.2.3.2 ENUMERATED

Purpose

A VARIABLE of type ENUMERATED is an unsigned integer. An associated list of triplets (integer, description, help) specifies the possible values of the VARIABLE plus a description and an optional help.

Lexical structure

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

The attributes of the ENUMERATED type are specified in Table 145.

Table 145 – Enumerated types attributes

Usage	Attribute	Description
m	description	(see Table 141)
m	value	is a constant that specifies the VARIABLE value. The type is an unsigned integer. The enumeration list is optional but if defined a value and a description is required. Equal values are not allowed
o	DEFAULT_VALUE	(see Table 141)
o	help	(see Table 141)
o	INITIAL_VALUE	(see Table 141)
o	size	(see Table 141)

7.27.2.2.3.3 INDEX

Purpose

A VARIABLE of TYPE INDEX is an unsigned integer which is used to reference the elements of a REFERENCE_ARRAY (see 7.22). The integer attribute of ELEMENTS (see 7.22.2.1) specifies the allowable values of the VARIABLE.

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

Lexical structure

INDEX (reference, size)<exp>

The attributes of the index type are specified in Table 146.

Table 146 – Index type attributes

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

7.27.2.2.4 Object reference types

Purpose

VARIABLEs of the type OBJECT_REFERENCE allow the access to values of other devices.

Lexical structure

OBJECT_REFERENCE DEFAULT_REFERENCE

The attributes of the OBJECT_REFERENCE types are specified in Table 147.

Table 147 – Object reference type attribute

Usage	Attribute	Description
o	DEFAULT_REFERENCE	specifies that the VARIABLE is used to access the values of other devices. If no DEFAULT_REFERENCE is defined the VARIABLE is initialized with SELF (see Table 148)

The attributes of the DEFAULT-REFERENCE are specified in Table 148.

Table 148 – DEFAULT_REFERENCE attributes

Usage	Attribute	Description
s	CHILD	navigates one hierarchy downward
s	FIRST	navigates to the first object in the hierarchy
s	LAST	navigates to the last object in the hierarchy
s	NEXT	navigates to the next object of the same hierarchy
s	PARENT	navigates one hierarchy bottom-up
s	PREV	navigates to the previous object of the same hierarchy
s	reference	is a reference to a device projected in the network
s	SELF	navigates back to themselves in their own hierarchy
s	NULL	is an empty object reference

7.27.2.2.5 ASCII, BIT_STRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE

Purpose

String variable types include the following.

- ASCII – is for specifying a sequence of characters from the ISO Latin-1 character set.
- BITSTRING – Variables of data type BIT_ENUMERATED are string constants which identify the position of a character of the VARIABLE value with its position.
- EUC (Extended Unit code) – is the internal code for specific characters (for example, China: EUC-CN, Taiwan EUC-TW). EUC is used to handle East Asian languages (ISO/IEC 10646-1).
- The EDDL processor shall support an encoding scheme where characters are encoded using a variable number of octets. Typically certain octets signal the beginning of a character and how many additional octets are used to encode the character. Character sets with a large number of elements are often stored using a packing scheme.
- PACKED_ASCII – is a subset of ASCII produced by removing the two most significant bits of each ASCII character (see D.5.7).
- PASSWORD – is a string type and is intended for specifying password strings. Except for how the variable is presented to the user, password and ASCII string types are identical.
- VISIBLE – This string is an ordered sequence of characters from ISO/IEC 10646-1 and ISO 2375 character set.
- OCTET – is for specifying a sequence of unformatted binary data whose definition is determined by the implementation of the device.

Lexical structure — EUC

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

Lexical structure — ASCII

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

Lexical structure — PACKED_ASCII

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

Lexical structure — PASSWORD

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

Lexical structure — BITSTRING

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

Lexical structure — VISIBLE

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

Lexical structure — OCTET

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

The attributes of the string types are specified in Table 149.

Table 149 – String types attributes

Usage	Attribute	Description
o	size	(see Table 141)
o	DEFAULT_VALUE	(see Table 141)
o	INITIAL_VALUE	(see Table 141)
o	string	is a string constant. The enumeration list is optional but if defined a string and a description is required. Equal strings are not allowed.
o	description	(see Table 141)
o	help	(see Table 141)

7.27.2.3 CONSTANT_UNIT**Purpose**

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

Lexical structure

CONSTANT_UNIT (string)<exp>

The attribute of CONSTANT_UNIT is specified in Table 150.

Table 150 – CONSTANT_UNIT attribute

Usage	Attribute	Description
m	string	specifies the units code string to be displayed

7.27.2.4 DEFAULT_VALUE**Purpose**

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

Lexical structure

DEFAULT_VALUE (expression)<exp>

NOTE The type of the expression must be consistent with the type of the VARIABLE.

NOTE The DEFAULT_VALUE may also be specified via the TYPE attribute of the VARIABLE.

The attribute of DEFAULT_VALUE is specified in Table 151.

Table 151 – DEFAULT_VALUE attribute

Usage	Attribute	Description
m	expression	specifies the default value of the variable

7.27.2.5 HANDLING

Purpose

The HANDLING attribute specifies the operations which can be performed on the VARIABLE. By default, a VARIABLE without a HANDLING attribute can be read and written.

Lexical structure

HANDLING (READ, READ_WRITE, WRITE) <exp>

The attributes of HANDLING are specified in Table 152.

Table 152 – HANDLING attribute

Usage	Attribute	Description
s	READ	specifies that the value of a VARIABLE may be read
s	READ_WRITE	specifies that the value of a VARIABLE may be read and written
s	WRITE	specifies that the value of a VARIABLE may be written

7.27.2.6 INITIAL_VALUE

Purpose

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

Lexical structure

INITIAL_VALUE expression

NOTE 1 The type of the expression must be consistent with the type of the VARIABLE.

NOTE 2 The INITIAL_VALUE may also be specified via the TYPE attribute of the VARIABLE.

The attribute of INITIAL_VALUE is specified in Table 150.

Table 153 – INITIAL_VALUE attribute

Usage	Attribute	Description
m	expression	specifies the initial value of the variable

7.27.2.7 POST_EDIT_ACTIONS

Purpose

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

Lexical structure

POST_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of POST_EDIT_ACTIONS are specified in Table 154.

Table 154 – POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS attributes

Usage	Attribute	Description
o	reference	is a reference to a METHOD instance
o	DEFINITION	lexical element (see 7.30.1)

7.27.2.8 POST_READ_ACTIONS

Purpose

The POST_READ_ACTIONS attribute specifies METHODS that shall be executed after the VARIABLE was read from the device. The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the following METHODS are not executed.

Lexical structure

POST_READ_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of POST_READ_ACTIONS are specified in Table 154.

7.27.2.9 POST_WRITE_ACTIONS

Purpose

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

Lexical structure

POST_WRITE_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of POST_WRITE_ACTIONS are specified in Table 154.

7.27.2.10 PRE_EDIT_ACTIONS

Purpose

The PRE_EDIT_ACTIONS attribute specifies METHODS that shall be executed immediately when the VARIABLE is displayed. 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

PRE_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of PRE_EDIT_ACTIONS are specified in Table 154.

7.27.2.11 PRE_READ_ACTIONS

Purpose

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

Lexical structure

PRE_READ_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of PRE_READ_ACTIONS are specified in Table 154.

7.27.2.12 PRE_WRITE_ACTIONS

Purpose

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

Lexical structure

PRE_WRITE_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of PRE_WRITE_ACTIONS are specified in Table 154.

7.27.2.13 REFRESH_ACTIONS

Purpose

The REFRESH_ACTIONS attribute specifies METHODS that shall be executed whenever the VARIABLE is displayed or refreshed. 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

REFRESH_ACTIONS [(reference)<exp>]+, DEFINITION

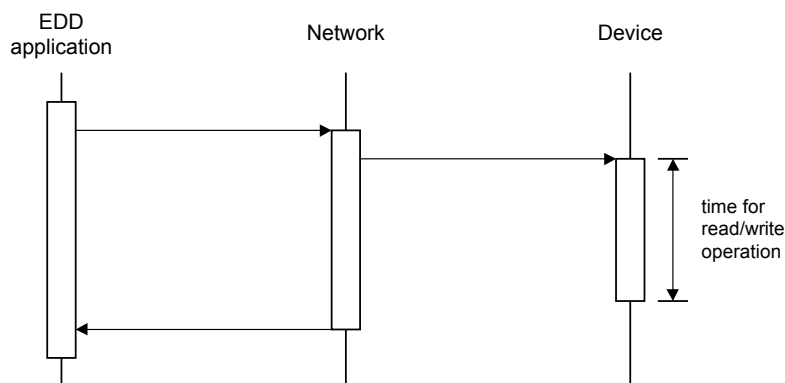
The attributes of REFRESH_ACTIONS are specified in Table 154.

7.27.2.14 READ_TIMEOUT, WRITE_TIMEOUT

Purpose

A READ_TIMEOUT specifies the length of time, in ms, the EDD application shall wait for the returned VARIABLE. Similarly, a WRITE_TIMEOUT specifies the length of time, in ms, an EDD application shall wait for confirmation that the VARIABLE is successfully written to the device.

For example, a WRITE_TIMEOUT should indicate the length of time it takes if a device stores the value of a variable.



IEC 1544/06

Figure 35 – Time for read-and-write operation

Lexical structure

```
READ_TIMEOUT (integer)<exp>
WRITE_TIMEOUT (integer)<exp>
```

The attribute of READ_TIMEOUT and WRITE_TIMEOUT is specified in Table 155.

Table 155 – READ/WRITE_TIMEOUT attributes

Usage	Attribute	Description
m	integer	specifies the time period in ms that the EDD application waits for a device to complete its requested update

7.27.2.15 STYLE

Purpose

The STYLE attribute specifies the way a VARIABLE is displayed. It is an additional information which is individually interpreted by the EDDL application.

Lexical structure

```
STYLE (string)<exp>
```

The attribute of STYLE is specified in Table 156.

Table 156 – STYLE attribute

Usage	Attribute	Description
m	string	specifies a string which is used for display purposes

7.27.2.16 VALIDITY

Purpose

The VALIDITY attribute specifies whether a VARIABLE is valid or invalid. A VARIABLE is invalid only if the VARIABLE does not exist. If the value is simply bad it is still valid. A VARIABLE without a VALIDITY is always valid.

NOTE 1 VALIDITY should be expressed using a conditional (IF, IF-ELSE, SELECT).

NOTE 2 If a VARIABLE is invalid, it should not be displayed.

Lexical structure

```
VALIDITY [(FALSE, TRUE)<exp>]
```

The attributes of VALIDITY are specified in Table 157.

Table 157 – VALIDITY attributes

Usage	Attribute	Description
s	FALSE	specifies that the VARIABLE instance is invalid
s	TRUE	specifies that the VARIABLE instance is valid

7.28 VARIABLE_LIST

Purpose

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

NOTE A VARIABLE_LIST may be transferred as one communication object.

Lexical structure

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

The attributes are specified in Table 158.

Table 158 – VARIABLE_LIST attributes

Usage	Attribute	Description
m	MEMBERS	lexical element (see 7.30.4)
o	HELP	lexical element (see 7.30.2)
o	LABEL	lexical element (see 7.30.3)
o	RESPONSE_CODES	lexical element (see 7.30.5)

7.29 WAVEFORM

7.29.1 General structure

Purpose

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

Lexical structure

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

The attributes of WAVEFORM are specified in Table 159.

Table 159 – WAVEFORM attributes

Usage	Attribute	Description
m	TYPE	lexical element (see 7.29.2.1)
o	EMPHASIS	lexical element (see 7.29.2.2)
o	EXIT_ACTIONS	lexical element (see 7.29.2.3)
o	HANDLING	lexical element (see 7.29.2.4)
o	HELP	lexical element (see 7.30.2)
o	INIT_ACTIONS	lexical element (see 7.29.2.5)
o	KEY_POINTS	lexical element (see 7.29.2.6)
o	LABEL	lexical element (see 7.30.3)
o	LINE_COLOR	lexical element (see 7.29.2.7)
o	LINE_TYPE	lexical element (see 7.29.2.8)
o	REFRESH_ACTIONS	lexical element (see 7.29.2.9)
o	Y_AXIS	lexical element (see 7.29.2.10)

7.29.2 Specific attributes

7.29.2.1 TYPE

7.29.2.1.1 General structure

Purpose

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

Lexical structure

TYPE [XY, YT, HORIZONTAL, VERTICAL]

The attributes of TYPE are specified in Table 160.

Table 160 – TYPE attribute

Usage	Attribute	Description
s	XY	a list of (x,y) points (see 7.29.2.1.2.1)
s	YT	a list of (y, t) points (see 7.29.2.1.2.2)
s	HORIZONTAL	a list of horizontal lines (see 7.29.2.1.2.3)
s	VERTICAL	a list of vertical lines (see 7.29.2.1.2.4)

7.29.2.1.2 Specific attributes**7.29.2.1.2.1 XY****Purpose**

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

Lexical structure

XY [X_VALUES, Y_VALUES, NUMBER_OF_POINTS]

The attributes of XY are specified in Table 161.

Table 161 – XY attribute

Usage	Attribute	Description
m	X_VALUES	Purpose The X_VALUES attribute specifies the X coordinate of each point in the WAVEFORM. For each X coordinate specified in X_VALUES there shall be a corresponding Y coordinate specified in Y_VALUES. Lexical Structure X_VALUES [reference, double, float, integer, unsigned_integer]+ The attributes of X_VALUES are specified in Table 169.
m	Y_VALUES	Purpose The Y_VALUES attribute specifies the Y coordinate of each point in the WAVEFORM. For each Y coordinate specified in Y_VALUES there shall be a corresponding X coordinate specified in X_VALUES. Lexical Structure Y_VALUES [(reference, double, float, integer, unsigned_integer)+ The attributes of Y_VALUES are specified in Table 169.
o	NUMBER_OF_POINTS	Purpose The NUMBER_OF_POINTS attribute specifies the number of valid data points in X_VALUES and Y_VALUES. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES and Y_VALUES. Lexical Structure NUMBER_OF_POINTS (integer)<exp>.

7.29.2.1.2.2 YT**Purpose**

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

Lexical structure

YT [X_INITIAL, X_INCREMENT, Y_VALUES, NUMBER_OF_POINTS]

The attributes of YT are specified in Table 162.

Table 162 – YT attribute

Usage	Attribute	Description
m	X_INITIAL	Purpose The X_INITIAL attribute specifies the X coordinate of the first point in the WAVEFORM. Lexical Structure X_INITIAL (reference)<exp> X_INITIAL (double)<exp> X_INITIAL (float)<exp> X_INITIAL (integer)<exp> X_INITIAL (unsigned_integer)<exp> The reference is a reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance or an element of a REFERENCE_ARRAY instance.
m	X_INCREMENT	Purpose The X_INCREMENT attribute specifies difference between the X coordinates of adjacent points in the WAVEFORM. Lexical Structure X_INCREMENT (reference)<exp> X_INCREMENT (double)<exp> X_INCREMENT (float)<exp> X_INCREMENT (integer)<exp> X_INCREMENT (unsigned_integer)<exp> The reference is a reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance or an element of a REFERENCE_ARRAY instance.
m	Y_VALUES	Purpose The Y_VALUES attribute specifies the Y coordinate of each point in the WAVEFORM. Lexical Structure Y_VALUES [(reference, double, float, integer, unsigned_integer)<exp>]+ The attributes of Y_VALUES are specified in Table 169.
o	NUMBER_OF_POINTS	Purpose The NUMBER_OF_POINTS attribute specifies the number of valid data points in X_VALUES and Y_VALUES. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES and Y_VALUES. Lexical Structure NUMBER_OF_POINTS (integer)<exp>.

7.29.2.1.2.3 HORIZONTAL**Purpose**

The HORIZONTAL attribute specifies a WAVEFORM that contains horizontal lines.

Lexical structure

HORIZONTAL Y_VALUES

The attributes of HORIZONTAL are specified in Table 163.

Table 163 – HORIZONTAL attribute

Usage	Attribute	Description
m	Y_VALUES	Purpose The Y_VALUES attribute specifies the Y coordinate of each horizontal line in the WAVEFORM. Lexical Structure Y_VALUES [reference, double, float, integer, unsigned_integer]+ The attributes of Y_VALUES are specified in Table 169.

7.29.2.1.2.4 VERTICAL**Purpose**

The VERTICAL attribute specifies a WAVEFORM that contains vertical lines.

Lexical structure

VERTICAL X_VALUES

The attributes of VERTICAL are specified in Table 164.

Table 164 – VERTICAL attribute

Usage	Attribute	Description
m	X_VALUES	Purpose The X_VALUES attribute specifies the X coordinate of each vertical line in the WAVEFORM. Lexical Structure X_VALUES [reference, double, float, integer, unsigned_integer]+ The attributes of X_VALUES are specified in Table 169.

7.29.2.2 EMPHASIS**Purpose**

The EMPHASIS attribute specifies whether or not the WAVEFORM should be emphasized when it is displayed. The way in which a WAVEFORM is emphasized is defined by the EDD application. By default, a WAVEFORM without an EMPHASIS attribute should not be emphasized.

Lexical structure

EMPHASIS (FALSE, TRUE) <exp>

The attributes of EMPHASIS are specified in Table 165.

Table 165 – EMPHASIS attribute

Usage	Attribute	Description
s	FALSE	specifies that the WAVEFORM should not be emphasized
s	TRUE	specifiers that the WAVEFORM should be emphasized

7.29.2.3 EXIT_ACTIONS**Purpose**

EXIT_ACTIONS are executed when the GRAPH containing the WAVEFORM is closed.

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

Lexical structure

EXIT_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of EXIT_ACTIONS are specified in Table 167.

7.29.2.4 HANDLING**Purpose**

The HANDLING attribute specifies the operations which can be performed on the WAVEFORM. By default, a WAVEFORM without a HANDLING attribute can be read and written.

Lexical structure

HANDLING (READ, READ_WRITE, WRITE)<exp>

The attributes of HANDLING are specified in Table 166.

Table 166 – HANDLING attribute

Usage	Attribute	Description
s	READ	specifies that the WAVEFORM may only be read
s	READ_WRITE	specifies that the WAVEFORM may be read and written
s	WRITE	specifies that the WAVEFORM may be written

7.29.2.5 INIT_ACTIONS**Purpose**

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

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

Lexical structure

INIT_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of INIT_ACTIONS are specified in Table 167.

Table 167 – EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS attribute

Usage	Attribute	Description
o	reference	is a reference to a METHOD instance
o	DEFINITION	lexical element (see XXX)

7.29.2.6 KEY_POINTS**7.29.2.6.1 General structure****Purpose**

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

Lexical structure

KEY_POINTS [X_VALUES, Y_VALUES]

The attributes of KEY_POINTS are specified in Table 168.

Table 168 – KEY_POINTS attribute

Usage	Attribute	Description
m	X_VALUES	specifies the X coordinate of each KEY_POINT
m	Y_VALUES	specifiers the Y coordinate of each KEY_POINT

7.29.2.6.2 Specific attributes**7.29.2.6.2.1 X_VALUES****Purpose**

The X_VALUES attribute specifies the X coordinate of each point that is to be highlighted. For each X coordinate specified in X_VALUES there shall be a corresponding Y coordinate specified in Y_VALUES.

Lexical structure

X_VALUES [reference, double, float, integer, unsigned_integer] +

The attributes of X_VALUES are specified in Table 169.

Table 169 – X_VALUES, Y_VALUES attribute

Usage	Attribute	Description
o	reference	is a reference to either a VARIABLE instance, a member of a RECORD instance, an element of a VALUE_ARRAY instance, an element of a REFERENCE_ARRAY instance, a VALUE_ARRAY instance or a REFERENCE_ARRAY instance
o	double	is a double-precision floating-point constant
o	float	is a single-precision floating-point constant
o	integer	is an integer constant
o	unsigned_integer	is an unsigned integer constant

7.29.2.6.2.2 Y_VALUES**Purpose**

The Y_VALUES attribute specifies the Y coordinate of each point that is to be highlighted. For each Y coordinate specified in Y_VALUES, there shall be a corresponding X coordinate specified in X_VALUES.

Lexical structure

Y_VALUES [reference, double, float, integer, unsigned_integer] +

The attributes of Y_VALUES are specified in Table 169.

7.29.2.7 LINE_COLOR**Purpose**

The LINE_COLOR attribute specifies the color that should be used to display the WAVEFORM. By default, a WAVEFORM without a LINE_COLOR attribute will be displayed using a default color defined by the EDD application.

Lexical structure

LINE_COLOR (integer, reference)<exp>

Colors are specified via a RGB value, as defined in Cascading Style Sheets, Level 2.

The attributes of LINE_COLOR are specified in Table 170.

Table 170 – LINE_COLOR attribute

Usage	Attribute	Description
s	integer	is the color to be used to display the WAVEFORM
s	reference	is a reference to a VARIABLE instance containing the color to be used to display the WAVEFORM

7.29.2.8 LINE_TYPE

Purpose

The LINE_TYPE attribute specifies the type of data contained in the WAVEFORM. Every WAVEFORM with the same LINE_TYPE attribute should be displayed in the same fashion (i.e., line color, line pattern, line thickness, etc). By default, a WAVEFORM without a LINE_TYPE attribute uses the DATA line style.

Lexical structure

LINE_TYPE ([DATA, n]*, LOW_LOW_LIMIT, LOW_LIMIT, HIGH_LIMIT, HIGH_HIGH_LIMIT)<exp>

The attributes of LINE_TYPE are specified in Table 171.

Table 171 – LINE_TYPE attribute

Usage	Attribute	Description
s	DATA	specifies the WAVEFORM contains data
c	n	is an increasing integer starting with one, which identifies distinct DATA line types if more than one is present
s	LOW_LOW_LIMIT	specifies the WAVEFORM contains a low low limit
s	LOW_LIMIT	specifies the WAVEFORM contains a low limit
s	HIGH_LIMIT	specifies the WAVEFORM contains a high limit
s	HIGH_HIGH_LIMIT	specifies the WAVEFORM contains a high high limit
s	TRANSPARENT	specifies that only the WAVEFORM data points are visible

7.29.2.9 REFRESH_ACTIONS

Purpose

REFRESH_ACTIONS are executed whenever changes are made to the X_AXIS or Y_AXIS of the GRAPH in which the WAVEFORM is being displayed.

The specified METHODS shall be executed in the order they appear. If a METHOD exits for an unplanned reason, the remaining METHODS shall not be executed. Furthermore, the X_VALUES and Y_VALUES of the WAVEFORM shall be restored to their original values by the EDD application, and the WAVEFORM on the GRAPH shall not be redrawn.

Lexical structure

REFRESH_ACTIONS [(reference)<exp>]+, DEFINITION

The attributes of REFRESH_ACTIONS are specified in Table 167.

7.29.2.10 Y_AXIS

Purpose

The Y_AXIS attribute specifies the way the Y axis of the WAVEFORM is displayed. Each WAVEFORM on a GRAPH share a common X_AXIS but each WAVEFORM on a GRAPH has its own Y_AXIS. By default, the display of the Y axis of a GRAPH without a Y_AXIS attribute is inferred from the data in the WAVEFORM.

Lexical structure

Y_AXIS (reference)<exp>

The attribute of Y_AXIS is specified in Table 172.

Table 172 – Y_AXIS attribute

Usage	Attribute	Description
m	reference	is a reference to an AXIS instance

7.30 Common attributes

7.30.1 DEFINITION

Purpose

A DEFINITION specifies actions to be performed and is based on a procedural programming language (for example, ANSI C). In the DEFINITIONS, METHODS and Builtins (see Annex B) may be called. Within a DEFINITION EDD basic elements can be used.

The following items shall be supported

- Basic data types (see D.5.1)
 - Arrays (see D.5.1)
 - Arithmetic operators (see D.5.1)
 - Statements (see D.5.1)
 - 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 effect the original data.

Lexical structure

DEFINITION compound-statement

The attribute of DEFINITION is specified in Table 173.

Table 173 – DEFINITION attributes

Usage	Attribute	Description
o	compound-statement	specifies a block of instructions for a supported procedural programming language

7.30.2 HELP

Purpose

HELP specifies text, which provides a description of the construct.

NOTE This text is intended to be used by EDD applications for user information.

Lexical structure

HELP (string)<exp>

The attribute of HELP is specified in Table 174.

Table 174 – HELP attribute

Usage	Attribute	Description
m	string	specifies user readable text

7.30.3 LABEL

Purpose

LABEL specifies the displayed designation of an element.

Lexical structure

LABEL (string)<exp>

The attribute of LABEL is specified in Table 175.

Table 175 – LABEL attribute

Usage	Attribute	Description
m	string	specifies user readable text

7.30.4 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).

Lexical structure

MEMBERS [(identifier, reference, description, help)<exp>]+

The attributes of MEMBERS are specified in Table 176.

Table 176 – MEMBERS attributes

Usage	Attribute	Description
m	identifier	specifies the name by which the item may be referenced
m	reference	is the reference to an EDD item. The type of the reference depends on the basic constructs If MEMBERS is a CHART attribute, reference is a reference to a SOURCE instance If MEMBERS is a COLLECTION attribute, reference is a reference to a AXIS, BLOCK_A, BLOCK_B, CHART, COLLECTION, COMMAND, CONNECTION, DOMAIN, EDIT_DISPLAY, FILE, GRAPH, LIST, MENU, METHOD, PROGRAM, RECORD, REFERENCE_ARRAY, REFRESH, RESPONSE_CODES, SOURCE, UNIT, VALUE_ARRAY, VARIABLE, VARIABLE_LIST, WAVEFORM, or WRITE_AS_ONE instance If MEMBERS is a FILE or LIST attribute, reference is a 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 If MEMBERS is a GRAPH attribute, reference is a reference to a WAVEFORM instance If MEMBERS is a RECORDS attribute, reference is a reference to a VARIABLE instance If MEMBERS is a SOURCE attribute, reference is a reference to a VARIABLE instance or a member of a RECORD instance If MEMBERS is a VARIABLE_LISTS attribute, reference is a reference to a RECORDS, or VALUE_ARRAY or VARIABLE instance
o	description	provides a short description of the item
o	help	specifies help text for the item

7.30.5 RESPONSE_CODES

Purpose

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

Two lexical structures for the RESPONSE_CODES attribute are provided:

- the referenced RESPONSE_CODES contains a reference to a RESPONSE_CODES instance.
- the embedded RESPONSE_CODES allows to specify the RESPONSE_CODES elements directly at the element instance.

Lexical structure for referenced RESPONSE_CODES

RESPONSE_CODES (reference) <exp>

The attribute of the referenced RESPONSE_CODES is specified in Table 177.

Table 177 – RESPONSE_CODES attribute

Usage	Attribute	Description
m	reference	is a reference to a RESPONSE_CODES instance

Lexical structure for embedded RESPONSE_CODES

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

The attributes of embedded RESPONSE_CODES are specified in Table 129.

7.31 Output redirection (OPEN and CLOSE)

Purpose

The output of the EDDL processor can be written to output files. The OPEN keyword instructs the EDDL processor to create a separate output file for the EDD constructs and the CLOSE keyword to close the output file. An attempt to open an output file that is already open results in an error. Opening and closing the same output file more than once in an EDD causes the processor to completely overwrite the contents of the file each time it is opened.

- The OPEN keyword opens an output file in overwrite mode. Each time an OPEN keyword is processed, an output file is opened, and created if necessary, unless the file is already open, in which case an error results. All objects generated are written to all open files.
- The CLOSE keyword prevents further output to an open file. If a file is reopened after closing, then any following objects overwrite the file.

Lexical structure

```
OPEN filename, [(construct)<exp>]*
CLOSE filename
```

The attributes are specified in Table 178.

Table 178 – OPEN and CLOSE attributes

Usage	Attribute	Description
M	filename	is a string of letters or digits which provides the file system name
M	construct	a list of constructs which are written in a separate output file

7.32 Conditional expression

Purpose

Conditional expressions are used to declare alternative values or to calculate values for an attribute of a basic element. The value of the expression is evaluated during execution time. The IF or SELECT conditional can be used to enable or disable referenced instances of a list of ITEMS attribute of MENU, etc.

Three kinds of conditional expressions may be used.

- An IF conditional is used for specifying an attribute that has two alternative definitions. The expression specified in the IF conditional is evaluated. If the result is non-zero, the attribute is specified by a then-clause; otherwise, it is specified by an else-clause.
- A SELECT conditional is used for specifying an attribute that has several alternative definitions. The select-expression is evaluated and compared with every integer_n of CASE. If a match is found, the appropriate select-clause is valid. If no matching integer_n is found, the default-clause following DEFAULT is used.
- A primary, unary or binary expression (see 7.34.7). This kind of conditional expression can only be used for numeric values (e.g. DEFAULT_VALUE, MIN_VALUE, SLOT, INDEX).

EXAMPLE It is possible to define more than one value for a DEFAULT_VALUE, MIN_VALUE, etc. using IF- or SELECT-conditional.

Lexical structure for the IF conditional

IF (if-expression), (then-clause)+ ELSE (else-clause)+

The attributes are specified in Table 179.

Lexical structure for the SELECT conditional

SELECT select-expression, (CASE integer_n, select-clause)+, (DEFAULT, default-clause)

The attributes are specified in Table 179.

Table 179 – IF, SELECT conditional

Usage	Attribute	Description
m	if-expression	is evaluated to determine whether the then-clause or the else-clause is used to define an attribute. If the if-expression contains a reference to a VARIABLE instance, the value of the referenced VARIABLE is used
m	then-clause	is the definition for the attribute if the value of condition is non-zero. The clause structure depends on the attribute being defined. It can also take the form of another conditional
o	else-clause	is the definition for the attribute if the value of condition is zero. The clause structure depends on the attribute being defined. It can also take the form of another conditional
m	select-expression	is evaluated to determine which select-clause is used to define an attribute. If the select-expression contains a reference to a VARIABLE instance, the value of the referenced VARIABLE is used
m	integer_n	specifies a number which identifies the according CASE. If integer matches with integer_n, the according select_clause is used
m	select-clause	is the definition for the attribute for each CASE, and the DEFAULT definition. The structure of each clause depends on the attribute being defined. Regardless of the attribute, each clause can also take the form of another conditional
o	default-clause	is the definition for the DEFAULT definition. The structure of default-clause depends on the attribute being defined. Regardless of the attribute, each clause can also take the form of another conditional

If the

- then-clause,
- else-clause,
- select-clause,
- default-clause

contains references to METHOD instances or Builtins, the returned value of the referenced METHOD or Builtin shall be used within the clause. The return values shall be type-compatible.

7.33 Referencing**7.33.1 Referencing an EDD instance****Purpose**

References are used within an EDD to refer to other EDD items.

EXAMPLE A PRE_EDIT_ACTION of VARIABLE refer to METHOD instances defined elsewhere in the EDD.

Lexical structure

The attribute is specified in Table 180.

Table 180 – Referencing an EDD instance

Usage	Attribute	Description
m	identifier	is the identifier of an EDD instance

7.33.2 Referencing bits of a BIT ENUMERATED VARIABLE

Purpose

This referencing is used to access a bit of a BIT ENUMERATED VARIABLE instance.

Lexical structure

variable-reference, bit-mask

The attributes are specified in Table 181.

Table 181 – Referencing elements of VARIABLE

Usage	Attribute	Description
m	variable-reference	is a reference to a VARIABLE instance; the type of the VARIABLE instance shall be BIT ENUMERATED
m	member-identifier	is a bit-mask which identifies a single bit within the variable; this is the mask used to define the bit within the VARIABLE

7.33.3 Referencing members of a RECORD

Purpose

This referencing is used to access a member of a RECORD instance.

Lexical structure

record-reference, member-identifier

The attributes are specified in Table 182.

Table 182 – Referencing elements of RECORD

Usage	Attribute	Description
m	record-reference	is a reference to a RECORD instance
m	member-identifier	is the identifier of an item of MEMBERS

7.33.4 Referencing elements of a VALUE_ARRAY

Purpose

This referencing is used to access an element of a VALUE_ARRAY instance.

Lexical structure

value-array-reference, expression

The attributes are specified in Table 183.

Table 183 – Referencing elements of VALUE_ARRAY

Usage	Attribute	Description
m	value-array-reference	is a reference to a VALUE_ARRAY instance
m	expression	is an expression which shall be evaluated to a non-negative integer that is within the index range of the VALUE_ARRAY (for a description of expressions, see 7.35)

7.33.5 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 184.

Table 184 – Referencing members of COLLECTION

Usage	Attribute	Description
m	collection-reference	is a reference to a COLLECTION instance
m	member-identifier	is the identifier of an item of MEMBERS

7.33.6 Referencing elements of a REFERENCE_ARRAY**Purpose**

This referencing is used to access an element of a REFERENCE_ARRAY instance.

Lexical structure

reference-array-reference, expression

The attributes are specified in Table 185.

Table 185 – Referencing members of REFERENCE_ARRAY

Usage	Attribute	Description
m	reference-array-reference	is a reference to a REFERENCE_ARRAY instance
m	expression	is an expression, which shall be evaluate to a non-negative integer that is within the index range of the REFERENCE_ARRAY (description of expressions see 7.34.7).

7.33.7 Referencing members of a VARIABLE_LISTS**Purpose**

This referencing is used to access a member of a VARIABLE_LISTS instance.

Lexical structure

variable-list-reference, member-identifier

The attributes are specified in Table 186.

Table 186 – Referencing members of VARIABLE_LISTS

Usage	Attribute	Description
m	variable-list-reference	is a reference to a VARIABLE_LISTS instance
m	member-identifier	is the identifier of an item of MEMBERS

7.33.8 Referencing elements of BLOCK_A PARAMETERS**Purpose**

This referencing is used to access an element of the PARAMETERS attribute of a BLOCK_A instance.

Lexical structure

PARAM, member-identifier

The attributes are specified in Table 187.

Table 187 – Referencing members of a BLOCK_A PARAMETERS

Usage	Attribute	Description
m	member-identifier	is the identifier of an item of MEMBERS

7.33.9 Referencing elements of BLOCK_A PARAMETER_LISTS**Purpose**

This referencing is used to access an element of the PARAMETER_LISTS attribute of a BLOCK_A instance.

Lexical structure

PARAM_LIST, member-identifier

The attributes are specified in Table 188.

Table 188 – Referencing members of BLOCK_A PARAMETER_LISTS

Usage	Attribute	Description
m	member-identifier	is the identifier of an item of MEMBERS

7.33.10 Referencing elements of BLOCK_A LOCAL_PARAMETERS**Purpose**

This referencing is used to access an element of the LOCAL_PARAMETERS attribute of a BLOCK_A instance.

Lexical structure

LOCAL_PARAM, member-identifier

The attributes are specified in Table 189.

Table 189 – Referencing members of BLOCK_A LOCAL_PARAMETER

Usage	Attribute	Description
m	member-identifier	is the identifier of an item of MEMBERS

7.33.11 Referencing BLOCK_A CHARACTERISTICS

Purpose

This referencing is used to access the CHARACTERISTICS of a BLOCK_A instance.

Lexical structure

BLOCK, characteristics-identifier

The attribute us specified in Table 190.

Table 190 – Referencing BLOCK_A CHARACTERISTICS

Usage	Attribute	Description
m	characteristics-identifier	is the identifier of the CHARACTERISTICS reference

7.33.12 Referencing members of a FILE

Purpose

This referencing is used to access a member of a FILE instance.

Lexical structure

file-reference, member-identifier

The attributes are specified in Table 191.

Table 191 – Referencing members of FILE

Usage	Attribute	Description
m	file-reference	is a reference to a FILE instance
m	member-identifier	is the identifier of an item of MEMBERS

7.33.13 Referencing elements of a LIST

Purpose

This referencing is used to access an element of a LIST instance.

Lexical structure

list-reference, (expression, FIRST, LAST)

The attributes are specified in Table 192.

Table 192 – Referencing elements of LIST

Usage	Attribute	Description
m	list-reference	is a reference to a LIST instance
s	expression	is an expression which shall be evaluate to a non-negative integer that is within the index range of the LIST (description of expressions see 7.34.7)
s	FIRST	references the first element of the LIST
s	LAST	references the last element of the LIST

7.33.14 Referencing members of a CHART

Purpose

This referencing is used to access a member of a CHART instance.

Lexical structure

`chart-reference, member-identifier`

The attributes are specified in Table 193.

Table 193 – Referencing members of CHART

Usage	Attribute	Description
m	chart-reference	is reference to a CHART instance
m	member-identifier	is the identifier of an item of MEMBERS

7.33.15 Referencing members of a GRAPH

Purpose

This referencing is used to access a member of a GRAPH instance.

Lexical structure

`graph-reference, member-identifier`

The attributes are specified in Table 194.

Table 194 – Referencing members of GRAPH

Usage	Attribute	Description
m	graph-reference	is reference to a GRAPH instance
m	member-identifier	is the identifier of an item of MEMBERS

7.33.16 Referencing members of a SOURCE

Purpose

This referencing is used to access a member of a GRAPH instance.

Lexical structure

`source-reference, member-identifier`

The attributes are specified in Table 195.

Table 195 – Referencing members of SOURCE

Usage	Attribute	Description
m	source-reference	is a reference to a SOURCE instance
m	member-identifier	is the identifier of an item of MEMBERS

7.33.17 Referencing AXIS of a GRAPH, SOURCE, WAVEFORM

Purpose

This referencing is used to access an AXIS of a GRAPH, SOURCE or WAVEFORM instance.

Lexical structure

reference, (X_AXIS, Y_AXIS)

The attributes are specified in Table 196.

Table 196 – Referencing AXIS of a GRAPH, SOURCE, WAVEFORM

Usage	Attribute	Description
m	reference	is a reference to a GRAPH instance, a reference to a SOURCE instance, or a reference to a WAVEFORM instance
c/o	X_AXIS	c only used with GRAPH; Y_AXIS used in all other cases
c/o	Y_AXIS	c only used with SOURCE and WAVEFORM; X_AXIS used in all other cases

7.34 Strings

7.34.1 Specifying a string as a string literal

Purpose

A text string can be specified as a string literal (see A.2.3).

Lexical structure

string

The attribute is specified in Table 197.

Table 197 – string as a string literal

Usage	Attribute	Description
m	string	specifies a string literal

7.34.2 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 attributes is specified in Table 198.

Table 198 – string as a string variable

Usage	Attribute	Description
m	reference	is a reference to a VARIABLE instance

7.34.3 Specifying a string as an enumeration value

Purpose

An enumeration value string is a string associated with one of the values of an enumeration variable.

Lexical structure

reference, value

The attributes are specified in Table 199.

Table 199 – string as an enumeration value

Usage	Attribute	Description
m	reference	is a reference to a VARIABLE instance of TYPE ENUMERATION
m	value	is the corresponding value of the description text within the enumeration list

Example The following enumeration value string specifies the string associated with the value 4 of the variable units_code: units_code (4)

7.34.4 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 200.

Table 200 – String as a dictionary reference

Usage	Attribute	Description
m	reference	is a reference to one or more strings specified in a dictionary (see 7.36)

7.34.5 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 201.

Table 201 – Referencing HELP and LABEL attributes of EDD instances

Usage	Attribute	Description
m	reference	references an EDD instance which contains the attributes LABEL and/or HELP
s	HELP	supplies the HELP string of the instance
s	LABEL	supplies the LABEL string of the instance

7.34.6 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 through the EDD application.

EXAMPLE `variable_of_type_string = "Description"`

`"Electronic " + "Device " + variable_of_type_string` results in `"Electronic Device Description"`.

Lexical structure

`(string)<exp>, [(concatenation-operator, string)<exp>]+`

The attributes are specified in Table 202.

Table 202 – String operation

Usage	Attribute	Description
m	string	is either a strings literals, string variables, string enumeration values, dictionary references or HELP, LABEL references
o	concatenation-operator	specifies that the strings shall be concatenated

7.34.7 Prompt string formats

Purpose

Prompt strings can be use to embed formatted items into strings. These items are composed of an optional format specifier and a VARIABLE ID or reference.

Lexical structure

`format, (reference, id)`

`label, (reference, id)`

The attributes are specified in Table 203.

Table 203 – Format specifier

Usage	Attribute	Description
m	format	consists of the ANSI C (see ISO 9899) printf format string. If the format is not specified, the format of the referenced VARIABLE instance is used. If no format is specified, a default format is used based on the type of VARIABLE instance
m	label	specifies the LABEL of the VARIABLE instance
o/s	reference	specifies the value or label of the referenced VARIABLE instance
o/s	id	specifies the item IDs of the VARIABLES

7.35 Expression

7.35.1 General structure

Purpose

There are three types of expressions.

- Primary expressions include constants, parenthesis-enclosed expressions, VARIABLE references, index element accesses or access to the status of VARIABLE instances. These operations are left-associative, i.e., they are evaluated from left to right.

- Unary expressions include all operators evaluating a single operand. These operations are right-associative, i.e., they are evaluated from right to left.
- Binary expressions include all operators evaluating two operands. These operations are left-associative, i.e., they are evaluated from left to right.

7.35.2 Primary expressions

Table 204 below summarizes the primary expressions in the EDDL.

Table 204 – Primary expressions

Primary expression	Description
constant	is an expression whose value is identical to the value of the constant
parenthesized expression	is an expression whose value is identical to the value of the enclosed expression
variable reference	is an expression whose value is the value of the referenced VARIABLE
ARRAY_INDEX	is an expression whose value was taken from the index number of the referencing VALUE_ARRAY
attribute values of VARIABLES	is a reference consisting of the identifier of a VARIABLE instance and an attribute. The reference supplies the according value of the attribute (see Table 205 for the returned values)
CURRENT_ROLE	specifies a string constant which contains the current role. The value of CURRENT_ROLE shall be supported by the EDD application. The values for the string constant shall be defined by the ROLE attribute of MENU
PI	specifies the mathematical constant π
attribute values of an AXIS	is a reference to a AXIS instance and an attribute. The reference supplies the according value of the attributes (see Table 206 for the returned values). These attributes may only be read unless specified otherwise
attribute values of a LIST	is a reference to a LIST instance and an attribute. The reference supplies the according value of the attributes (see Table 207 for the returned values). These attributes may only be read unless specified otherwise

Table 205 – Attribute values of VARIABLES

Attributes	Data type	Description
MIN_VALUE	type of VARIABLE instance	returns a 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 141) shall be used for referencing
MAX_VALUE	type of VARIABLE instance	returns a 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 141) shall be used for referencing
SCALING_FACTOR	double	returns a value which complies with the SCALING_FACTOR of the VARIABLE instance
DEFAULT_VALUE	type of VARIABLE instance	returns a value which complies with the DEFAULT_VALUE of the VARIABLE instance
INITIAL_VALUE	type of VARIABLE instance	returns a value which complies with the INITIAL_VALUE of the VARIABLE instance
VARIABLE_STATUS		returns a value which indicates the status of VARIABLE. Following status 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 transferred

Attributes	Data type	Description
		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 successful 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 206 – AXIS Attribute Values

Attributes	Data type	Description
MIN_VALUE	arithmetic type	returns a value which complies with the MIN_VALUE of the AXIS instance
MAX_VALUE	arithmetic type	returns a value which complies with the MAX_VALUE of the AXIS instance
SCALING	LINEAR or LOGARITHMIC	returns a value which complies with the SCALING of the AXIS instance
VIEW_MIN	arithmetic type	Returns a value which complies with the VIEW_MIN of the AXIS instance; VIEW_MIN can be read and written
VIEW_MAX	arithmetic type	Returns a value which complies with the VIEW_MAX of the AXIS instance; VIEW_MAX can be read and written

Table 207 – LIST Attribute Values

Attributes	Data type	Description
CAPACITY	unsigned_integer	returns a value which complies with the CAPACITY of the LIST instance
COUNT	unsigned_integer	returns a value which complies with the COUNT of the LIST instance

7.35.3 Unary expressions

A unary expression consists of a numerical operand, an expression, preceded by a unary operator. Table 208 specifies the unary expressions in the EDDL.

Table 208 – Unary expressions

Operator	Description
++	increments its operand by adding the value 1 to the value of the operand. The increment operator can be used either as prefix operator or as postfix operator. Both the prefix and the postfix notation will cause an increment of the operand. The difference is that the prefix expression changes the operand before using its value, while the postfix expression changes the operand after using its value. Thus the result of these expression can have different meanings according to the program context
--	decrements its operand by subtracting the value 1 from the operand. The decrement operator can be used either as prefix operator or as postfix operator. Both the prefix and the postfix notation will cause a decrement of the operand. The difference is that the prefix expression changes the operand before using its value, while the postfix expression changes the operand after using its value. Thus, the result of these expressions can have different meanings according to the program context
-	is the arithmetic negation of its operand
~	is the 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
!	is the logical negation of its operand

7.35.4 Binary expressions

7.35.4.1 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. This subsection specifies the following types of binary expressions.

- Multiplicative
- Additive
- Shift
- Relational
- Equality
- Bitwise AND (&)
- Bitwise XOR (^)
- Bitwise OR (|)
- Logical AND (&&)
- Logical OR (||)
- Conditional evaluation
- Assignments

7.35.4.2 Multiplicative operators

Multiplicative operators specify multiplication and division of operands. Table 209 specifies the multiplicative operators.

Table 209 – Multiplicative operators

Operator	Description
*	specifies the multiplication of its operands
/	specifies the division of the first operand by the second operand. The result of the / operator is the quotient of the division
%	specifies the division of the first operand by the second operand. The result of the % operator is the remainder

7.35.4.3 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 subtraction. The resulting value of this operation is the difference of its operands. The second operand is subtracted from the first. Table 210 specifies additive operators.

Table 210 – Additive operators

Operator	Description
+	specifies the addition of its operands
-	specifies the subtraction of the second operand from the first

7.35.4.4 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 211 specifies the shift operators.

Table 211 – Shift operators

Operator	Description
<<	shifts the first operand to the left. The bits shifted off are discarded, and the vacated bits are zero filled
>>	shifts the first operand to the right. The bits shifted off are discarded. If the first operand is less than 0, the vacated bits are one filled; otherwise they are zero filled

7.35.4.5 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 212 specifies the relational operators.

Table 212 – Relational operators

Operator	Description
<	tests for the relationship "less than"
<=	tests for the relationship "less than or equal to"
>	tests for the relationship "greater than"
>=	tests for the relationship "greater than or equal to"

7.35.4.6 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 213 specifies the equality operators.

Table 213 – Equality operators

Operator	Description
==	tests for the relationship "equals"
!=	tests for the relationship "does not equal"

7.35.4.7 Bitwise AND operator (&)

The & operator specifies the bitwise AND of its integer operands, that is, each bit of the result is set if each of the corresponding bits of the operands is set. The operands of the & operator shall have integral values.

7.35.4.8 Bitwise XOR operator (^)

The ^ operator specifies the bitwise exclusive OR of its integer operands, that is, each bit of the result is set if only one of the corresponding bits of the operands is set. The operands of the ^ operator shall have integral values.

7.35.4.9 Bitwise OR operator (|)

The | operator specifies the bitwise inclusive OR of its integer operands, that is, each bit of the result is set if either of the corresponding bits of the operands is set. The operands of the | operator shall have integral values.

7.35.4.10 Logical AND operator (&&)

The && operator specifies the Boolean AND evaluation of its integer operands. The result of this type of expression is 1 if both of the operands are not equal to 0; otherwise, the result is 0. If the first operand is equal to 0, the second operand is not evaluated.

7.35.4.11 Logical OR operator (||)

The || operator specifies the Boolean OR evaluation of its integer operands. The result of this type of expression is 1 if either of the operands is not equal to 0; otherwise, the result is 0. If the first operand is not equal to 0, the second operand is not evaluated.

7.35.4.12 Conditional evaluation

The operators ? and : are used for conditional evaluation.

EXAMPLE 1

```
result = logexpr ? trueexpr : falseexpr;
logexpr is evaluated and if it is non-zero, the result is the value of the trueexpr. Otherwise the falseexpr is executed.
```

EXAMPLE 2 The following if-else structure is logical identical to example 1:

```
if (logexpr)
    result = trueexpr ;
else
    result = falseexpr ;
```

7.35.4.13 Assignments

Procedural programming languages provide a series of assignment operators, all of which are right-associative. All assignment operators require a unary expression as their left operand. The right operand can be an assignment expression again. The type of the assignment expression corresponds to its left operand. The value of an assignment operation is the value stored in the left operand after the assignment has taken place.

The binary = operator indicates the simple assignment. The binary operators *=, /=, %=, +=, -=, >>=, <<=, &=, ^=, and |= indicate a compound assignment expression.

A compound assignment of general form as in

```
expr1 <operator>= expr2
```

is equivalent with the expression

```
expr1 = expr1 <operator> (expr2)
```

The data type of expr2 shall be convertible to the data type of expr1. The semantics of the assignments are specified in 7.35.2, 7.35.3 and 7.35.4.

7.35.4.14 Expression list

Each expression can consist of a list of comma-separated binary expressions. A pair of expressions separated by a comma is evaluated left to right and the value of the left expression is discarded. The type and value of the result are the type and value of the right operand.

EXAMPLE 1 `c -= (a=5, b=3-a, c=b+a, a*=b+=4+2*a);`
where the values 60, 12, and -57 are stored to the variables a, b, and c, respectively.

EXAMPLE 2 `fct ( x, (y=8, y*25.4), z )`

has three arguments, the second of which has the value 203.2. In contexts where the comma is given a special meaning, for example, in a list of actual function parameters and lists of initializers, the comma operator can only appear in parentheses; for example, the function call

7.36 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

`identifier (language-code, country-code, string)+`

The attributes are specified in Table 214.

Table 214 – Text dictionary attributes

Usage	Attribute	Description
m	Identifier	is used to reference the text string in an EDD
m	language-code	specifies the language used and the ASCII or multi code presentation of a string; language codes are defined in ISO 639
o	country-code	specifies the the country in which the language is spoken; country codes are defined in ISO 3166
m	string	specifies the text string in the related language

In addition to the standard country codes defined in ISO 3166, 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

```
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 Spain. 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.

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 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 causes 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 processing parts of a 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` in 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 processing parts of an EDD file. The `#ifdef` and `#ifndef` directives perform the same task as the `#if` directive when it is used with 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 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 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 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 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 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	a non-empty sequence of the binary digits 0 and 1 preceded by either 0b or 0B
Octal	a non-empty sequence of the digits 0 through 7 beginning with 0
Decimal	a non-empty sequence of the decimal digits 0 through 9, not beginning with 0
Hexadecimal	a 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

A string literal is a possibly empty sequence of characters enclosed in double quotes ("). The enclosed characters shall be any ISO Latin-1 (ISO/IEC 8859-1) character except the following:

- double quote (");
- backslash (\);
- new line.

Using escape sequences in string literals a string constant can also contain escape sequences that represent an ISO Latin-1 character. Table A.2 shows the escape sequences and their result.

Table A.2 – Using escape sequences in string literals

Escape code	Result
'	single quote
"	double quote
	vertical bar
?	question mark
\	backslash
\a	alert
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

A.2.4 Using language codes in string constants

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 – Using language codes in string literals

Language Code	Language
no code assigned	default language, used if a language is requested that is not defined
cn	Chinese
de	German
en	English
es	Spanish
fr	French
it	Italian
jp	Japanese
kr	Korean

NOTE English is recommended as the default language.

Example The following string literal specifies the English phrase "Invalid Selection" in English and German:

"Invalid Selection"
 "|de|Unzulässige Auswahl"

A.3 Operators

Table A.4 contains operators of the EDDL.

Table A.4 – EDDL operators

!	!=	%	%=
&	&&	&=	(
)	*	*=	+
++	+=	,	-
--	-=	.	
/=	:	;	<
<<	<<=	<=	=
==	>	>=	>>
>>=	?	[	]
^	^=	{	
=		}	~
->			

A.4 Keywords

Table A.5 contains the EDDL keywords:

Table A.5 – EDDL keywords

EDDL keyword	EDDL keyword	EDDL keyword	EDDL keyword
!	COMM	HART	PROGRAMS
%	COMMAND	HEADER	PURPOSE
&	COMMANDS	HELP	READ
(	COMPUTATION	HIDDEN	READ_ONLY
)	CONNECTION	IF	READ_TIMEOUT
*	CONNECTIONS	if	RECORD
+	CONSTANT_UNIT	IGNORE_IN_TEMPORARY_MASTER	RECORDS
,	CONTAINED	IMPORT	REDEFINE
-	continue	INDEX	REDEFINE DOMAIN
.	CORRECTABLE	INFO	REDEFINITIONS
/	CORRECTION	INITIAL_VALUE	REFRESH
:	DATA	INPUT	REFRESH_ITEMS
;	DATA_ENTRY_ERROR	int	REFRESHES
<	DATA_ENTRY_WARNING	INTEGER	RELATIONS
=	DATA_EXCHANGE	ITEM_ARRAY	REPLY
>	DATE	ITEM_ARRAY_ITEMS	REQUEST
?	DATE_AND_TIME	ITEM_ARRAYS	RESPONSE_CODES
[	DD_REVISION	ITEMS	return
]	DEFAULT	LABEL	REVIEW
^	default	LAST	ROLE

EDDL keyword	EDDL keyword	EDDL keyword	EDDL keyword
{	DEFAULT_VALUE	LIKE	ROOT
	DEFINITION	LOAD_TO_APPLICATION	SCALING_FACTOR
}	DELETE	LOAD_TO_DEVICE	SELECT
~	DELETE DOMAIN	LOCAL	SELF
!=	DETAIL	LOCAL_DISPLAY	SELF_CORRECTING
%=	DEVICE	long	SERVICE
&&	DEVICE_REVISION	MANUFACTURER	short
&=	DEVICE_TYPE	MANUFACTURER_EXT	signed
*=	DIAGNOSE	MAX_VALUE	SLOT
++	DIAGNOSTIC	MEMBERS	SOFTWARE
+=	DIALOG	MENU	STATE
--	DIGITAL_INPUT	MENU_ITEMS	STYLE
-=	DIGITAL_OUTPUT	MENUS	SUCCESS
->	DISCRETE_INPUT	METHOD	SUMMARY
/=	DISCRETE_OUTPUT	METHOD_ITEMS	switch
<<	DISPLAY_FORMAT	METHODS	TABLE
<=	DISPLAY_ITEMS	MIN_VALUE	TIME
==	DISPLAY_VALUE	MISC	TIME_VALUE
>=	do	MISC_ERROR	TRANSACTION
>>	DOMAIN	MISC_WARNING	TRANSDUCER
\0	DOMAINS	MODE	TRUE
^=	DOUBLE	MODE_ERROR	TUNE
=	double	MODULE	TYPE
	DURATION	MORE	UNCORRECTABLE
<<=	DYNAMIC	NEXT	UNIT
>>=	EDD_PROFILE	NUMBER	UNIT_ITEMS
ACCESS	EDD_REVISION	NUMBER_OF_ELEMENTS	UNITS
ADD	EDD_VERSION	OBJECT_REFERENCE	unsigned
ALARM	EDIT_DISPLAY	OCTET	UNSIGNED_INTEGER
ANALOG_INPUT	EDIT_DISPLAY_ITEMS	OF	VALIDITY
ANALOG_OUTPUT	EDIT_DISPLAYS	OFFLINE	VARIABLE
APPINSTANCE	EDIT_FORMAT	ONLINE	VARIABLE_LIST
ARGUMENTS	EDIT_ITEMS	OPERATE	VARIABLE_LISTS
ARRAY	ELEMENTS	OPERATION	VARIABLES
ARRAYS	ELSE	OUTPUT	VARIABLE_STATUS
ASCII	else	PACKED_ASCII	VARIABLE_STATUS_NONE
BAD	ENTRY	PARAMETER_LISTS	VARIABLE_STATUS_INVALID
BIT_ENUMERATED	ENUMERATED	PARAMETERS	VARIABLE_STATUS_NOT_ACCEPTED
BITSTRING	EUC	PARENT	VARIABLE_STATUS_NOT_SUPPORTED
BLOCK	EVENT	PASSWORD	VARIABLE_STATUS_CHANGED
BLOCKS	EVERYTHING	PHYSICAL	VARIABLE_STATUS_LOADED
break	FALSE	POST_EDIT_ACTIONS	VARIABLE_STATUS_INITIAL
CASE	FIRST	POST_READ_ACTIONS	VARIABLE_STATUS_NOT_CONVERTED

EDDL keyword	EDDL keyword	EDDL keyword	EDDL keyword
			ERTED
case	FLOAT	POST_WRITE_ACTIONS	VISIBLE
char	float	PRE_EDIT_ACTIONS	while
CHARACTERISTICS	for	PRE_READ_ACTIONS	WINDOW
CHILD	FREQUENCY_INPUT	PREV	WRITE
CLASS	FREQUENCY_OUTPUT	PRE_WRITE_ACTIONS	WRITE_AS_ONE
COLLECTION	FUNCTION	PROCESS	WRITE_AS_ONE_ITEMS
COLLECTION_ITEMS	HANDLING	PROCESS_ERROR	WRITE_AS_ONES
COLLECTIONS	HARDWARE	PROGRAM	WRITE_TIMEOUT

A.5 Terminals

The following text contain 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           = { '\"?afnrty' } .
    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 * \"/>

```

A.6 letter (letter|digit|_)* Formal EDDL syntax

A.6.1 General

This annex contains a formal syntax of the EDDL using the Backus Naur Form.

If there are optional elements each element is marked with /* M */ if the element is mandatory or /* O */ if the element is optional.

A.6.2 EDD identification information

```

device_description
    = identification definition_list

identification
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision
    = manufacturer ',' device_type ',' device_revision ','

```

```

    DD_revision ',' edd_version
= manufacturer ',' device_type ',' device_revision ','
    DD_revision ',' edd_profile
= manufacturer ',' device_type ',' device_revision ','
    DD_revision ',' manufacturer_ext
= manufacturer ',' device_type ',' device_revision ','
    DD_revision ',' edd_version ',' edd_profile
= manufacturer ',' device_type ',' device_revision ','
    DD_revision ',' edd_version ',' manufacturer_ext
= manufacturer ',' device_type ',' device_revision ','
    DD_revision ',' edd_profile ',' manufacturer_ext
= manufacturer ',' device_type ',' device_revision ','
    DD_revision ',' edd_version ',' edd_profile ','
    manufacturer_ext

manufacturer
= 'MANUFACTURER' Integer
= 'MANUFACTURER' Identifier

device_type
= 'DEVICE_TYPE' Integer
= 'DEVICE_TYPE' Identifier

device_revision
= 'DEVICE_REVISION' Integer

DD_revision
= 'DD_REVISION' Integer
= 'EDD_REVISION' Integer

edd_version
= 'EDD_VERSION' Integer

edd_profile
= 'EDD_PROFILE' Integer

manufacturer_ext
= 'MANUFACTURER_EXT' string

definition_list
= definition_listR

definition_listR
= definition
= definition_listR definition

definition
= item
= imported_description
= like

item
= axis
= array
= block
= chart
= collection
= command
= connection
= domain
= edit_display
= file
= graph
= list
= item_array
= menu
= method

```

```
= program
= record
= refresh_relation
= response_codes_definition
= source
= unit_relation
= variable
= variable_list
= waveform
= wao_relation
```

A.6.3 AXIS

```
axis
= 'AXIS' Identifier '{' axis_attribute_list '}'
= 'AXIS' Identifier '{' '}'

axis_attribute_list
= axis_attribute_listR

axis_attribute_listR
= axis_attribute
= axis_attribute_listR axis_attribute

axis_attribute
= axis_label
= axis_min_value
= axis_max_value
= axis_scaling
= axis_help
= constant_unit

axis_min_value
= minimum_value

axis_max_value
= maximum_value

axis_label
= label

axis_scaling
= 'SCALING' axis_scaling_specifier

axis_scaling_specifier
= axis_scaling_keyword ';'
= 'IF' '(' expr ')' '{' axis_scaling_specifier '}'
= 'IF' '(' expr ')' '{' stmt1: axis_scaling_specifier '}'
'ELSE' '{' stmt2: axis_scaling_specifier '}'
= 'SELECT' '(' expr ')' '{' axis_scaling_selection_list '}'

axis_scaling_selection_list
= axis_scaling_selection_listR

axis_scaling_selection_listR
= axis_scaling_selection
= axis_scaling_selection_listR axis_scaling_selection

axis_scaling_selection
= 'CASE' expr ':' axis_scaling_specifier
= 'DEFAULT' ':' axis_scaling_specifier

axis_scaling_keyword
= 'LINEAR'
= 'LOGARITHETIC'

axis_help
= help
```

A.6.4 BLOCK_A and BLOCK_B

```

block
    = 'BLOCK' Identifier '{' block_attribute_list '}'

block_attribute_list
    = block_attribute
    = block_attribute_list block_attribute

block_attribute
    = block_a_axis_items                /* O */
    = block_a_characteristics            /* M */
    = 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 */
    = block_a_image_items                /* O */
    = block_a_help                       /* O */
    = block_a_item_array_items           /* O */
    = block_a_label                      /* M */
    = block_a_list_items                 /* O */
    = block_a_local_parameters           /* O */
    = block_a_menu_items                 /* O */
    = block_a_method_items               /* O */
    = block_a_parameters                 /* M */
    = block_a_parameter_lists            /* O */
    = block_a_refresh_items              /* O */
    = block_a_source_items               /* O */
    = block_a_unit_items                 /* O */
    = block_a_wao_items                  /* O */
    = block_a_waveform_items             /* O */

    = block_b_number                     /* M */
    = block_b_type                       /* M */

block_a_help
    = help

block_a_label
    = label

block_a_characteristics
    = 'CHARACTERISTICS' record_reference ';'

block_a_parameters
    = 'PARAMETERS' '{' member_list '}'

member_list
    = member
    = member_list member

block_a_parameter_lists
    = 'PARAMETER_LISTS' '{' 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_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_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_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_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_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_unit_items
    = 'UNIT_ITEMS' '{' unit_items_specifier_list '}'

unit_items_specifier_list
    = unit_items_specifier
    = unit_items_specifier_list ',' unit_items_specifier

unit_items_specifier
    = unit_reference

block_a_wao_items
    = 'WRITE_AS_ONE_ITEMS' '{' wao_items_specifier_list '}'

wao_items_specifier_list
    = wao_items_specifier
    = wao_items_specifier_list ',' wao_items_specifier

```

```

wao_items_specifier
    = wao_reference

block_a_axis_items
    = 'AXIS_ITEMS' '{' axis_items_specifier_list '}'

axis_items_specifier_list
    = axis_items_specifier_listR

axis_items_specifier_listR
    = axis_items_specifier
    = axis_items_specifier_listR ',' 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_listR

chart_items_specifier_listR
    = chart_items_specifier
    = chart_items_specifier_listR ',' chart_items_specifier

chart_items_specifier
    = chart_reference

block_a_file_items
    = 'FILE_ITEMS' '{' file_items_specifier_list '}'

file_items_specifier_list
    = file_items_specifier_listR

file_items_specifier_listR
    = file_items_specifier
    = file_items_specifier_listR ',' 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_listR

graph_items_specifier_listR
    = graph_items_specifier
    = graph_items_specifier_listR ',' 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_listR

grid_items_specifier_listR
    = grid_items_specifier
    = grid_items_specifier_listR ',' 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_listR

image_items_specifier_listR
    = image_items_specifier
    = image_items_specifier_listR ',' image_items_specifier

image_items_specifier
    = image_reference

block_a_list_items
    = 'LIST_ITEMS' '{' list_items_specifier_list '}'

list_items_specifier_list
    = list_items_specifier_listR

list_items_specifier_listR
    = list_items_specifier
    = list_items_specifier_listR ',' list_items_specifier

list_items_specifier
    = list_reference

block_a_source_items
    = 'SOURCE_ITEMS' '{' source_items_specifier_list '}'

source_items_specifier_list
    = source_items_specifier_listR

source_items_specifier_listR
    = source_items_specifier
    = source_items_specifier_listR ',' source_items_specifier

source_items_specifier
    = source_reference

block_a_waveform_items
    = 'WAVEFORM_ITEMS' '{' waveform_items_specifier_list '}'

waveform_items_specifier_list
    = waveform_items_specifier_listR

waveform_items_specifier_listR
    = waveform_items_specifier
    = waveform_items_specifier_listR ',' waveform_items_specifier

waveform_items_specifier
    = waveform_reference

block_a_local_parameters
    = 'LOCAL_PARAMETERS' '{' member_list '}'

block_b_number
    = 'NUMBER' expr ';'

block_b_type
    = 'TYPE' 'PHYSICAL' ';'
    = 'TYPE' 'TRANSDUCER' ';'
    = 'TYPE' 'FUNCTION' ';'

```

A.6.5 CHART

chart

```

    = 'CHART' Identifier '{' chart_attribute_list '}'

chart_attribute_list
    = chart_attribute_listR

chart_attribute_listR
    = chart_attribute
    = chart_attribute_listR chart_attribute

chart_attribute
    = chart_label                /* O */
    = chart_help                 /* O */
    = chart_members              /* M */
    = chart_height               /* O */
    = chart_width                /* O */
    = chart_type                 /* O */
    = chart_cycle_time           /* O */
    = chart_length               /* O */

chart_label
    = label

chart_help
    = help

chart_members
    = members

chart_height
    = 'HEIGHT' display_size_specifier

chart_width
    = 'WIDTH' display_size_specifier

chart_type
    = 'TYPE' chart_type_specifier

chart_type_specifier
    = chart_type_keyword ':'
    = 'IF' '(' expr ')' '{' chart_type_specifier '}'
    = 'IF' '(' expr ')' '{' stmt1: chart_type_specifier '}'
    'ELSE' '{' stmt2: chart_type_specifier '}'
    = 'SELECT' '(' expr ')' '{' chart_type_selection_list '}'

chart_type_selection_list
    = chart_type_selection_listR

chart_type_selection_listR
    = chart_type_selection
    = chart_type_selection_listR 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'

```

```
chart_cycle_time
    = 'CYCLE_TIME' expr_specifier
```

```
chart_length
    = 'LENGTH' expr_specifier
```

A.6.6 COLLECTION

```
collection
    = 'COLLECTION' Identifier '{' collection_attribute_list '}'
    = 'COLLECTION' 'OF' collection_item_type Identifier '{'
        collection_attribute_list '}'
```

```
collection_item_type
    = item_type
```

```
item_type
    = 'ARRAY'
    = 'AXIS'
    = 'BLOCK'
    = 'CHART'
    = 'COLLECTION' 'OF' item_type
    = 'COMMAND'
    = 'CONNECTION'
    = 'DOMAIN'
    = 'EDIT_DISPLAY'
    = 'FILE'
    = 'GRAPH'
    = 'ITEM_ARRAY' 'OF' item_type
    = 'LIST'
    = 'MENU'
    = 'METHOD'
    = 'PROGRAM'
    = 'RECORD'
    = 'REFRESH'
    = 'RESPONSE_CODES'
    = 'SOURCE'
    = 'UNIT'
    = 'VARIABLE'
    = 'VARIABLE_LIST'
    = 'WAVEFORM'
    = 'WRITE_AS_ONE'
```

```
collection_attribute_list
    = collection_attribute
    = collection_attribute_list collection_attribute
```

```
collection_attribute
    = help                /* O */
    = label                /* O */
    = members              /* M */
```

A.6.7 COMMAND

```
command
    = 'COMMAND' Identifier '{' command_attribute_list '}'
    = 'COMMAND' Identifier '{' '}'
```

```
command_attribute_list
    = command_attribute
    = command_attribute_list command_attribute
```

```
command_attribute
    = command_header      /* O */
    = command_slot         /* O */
    = command_index        /* O */
    = command_block        /* O */
```

```

= command_number          /* O */
= command_operation        /* M */
= command_transaction      /* M */
= command_connection       /* O */
= command_module           /* O */
= command_response_codes  /* O */

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

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_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_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' number_specifier

number_specifier:
    = number_items ';'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    'ELSE' '{' number_specifier '}'
    = 'SELECT' '(' expr ')' '{' number_selection_list '}'

number_items:
    = Integer
    = variable_reference

number_selection_list
    = number_selection
    = number_selection_list number_selection

number_selection
    = 'CASE' expr ':' number_specifier
    = 'DEFAULT' ':' number_specifier

command_operation
    = 'OPERATION' operation_specifier

operation_specifier
    = operation ':'
    = 'IF' '(' expr ')' '{' operation_specifier '}'
    = 'IF' '(' expr ')' '{' operation_specifier '}'
    'ELSE' '{' operation_specifier '}'
    = 'SELECT' '(' expr ')' '{' operation_selection_list '}'

operation_selection_list
    = operation_selection
    = operation_selection_list operation_selection

operation_selection
    = 'CASE' expr ':' operation_specifier
    = 'DEFAULT' ':' operation_specifier

operation
    = 'READ'
    = 'WRITE'
    = 'COMMAND'
    = 'DATA_EXCHANGE'
    = string

```

```

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 */

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
    = data_items
    = data_items_list ',' data_items

data_items
    = Integer
    = variable_reference
    = variable_reference '<' Integer '>'
    = variable_reference '(' data_items_qualifiers ')'
    = variable_reference '<' Integer '>' '(' data_items_qualifiers ')'

data_items_qualifiers
    = data_items_qualifiers_wrap

data_items_qualifiers_wrap
    = data_items_qualifier
    = data_items_qualifiers_wrap ',' data_items_qualifier

data_items_qualifier
    = 'INDEX'
    = 'INFO'

command_connection
    = 'CONNECTION' connection_specifier

connection_specifier
    = connection_reference ';'
    = 'IF' '(' expr ')' '{' connection_specifier '}'
    = 'IF' '(' expr ')' '{' connection_specifier '}'
    = 'ELSE' '{' connection_specifier '}'
    = 'SELECT' '(' expr ')' '{' connection_selection_list '}'

connection_selection_list
    = connection_selection
    = connection_selection_list connection_selection

connection_selection
    = 'CASE' expr ':' connection_specifier
    = 'DEFAULT' ':' connection_specifier

command_module
    = 'MODULE' module_specifier

```

```

module_specifier
= Identifier ';'
= 'IF' '(' expr ')' '{' module_specifier '}'
= 'IF' '(' expr ')' '{' module_specifier '}'
  'ELSE' '{' module_specifier '}'
= 'SELECT' '(' expr ')' '{' module_selection_list '}'

module_selection_list
= module_selection
= module_selection_list module_selection

module_selection
= 'CASE' expr ':' module_specifier
= 'DEFAULT' ':' module_specifier

command_response_codes
= response_codes

```

A.6.8 CONNECTION

```

connection
= 'CONNECTION' Identifier '{' connection_attribute_list '}'

connection_attribute_list
= connection_attribute
= connection_attribute_list connection_attribute

connection_attribute
= appinstance /* M */

appinstance
= 'APPINSTANCE' Integer ';'

```

A.6.9 DOMAIN

```

domain
= 'DOMAIN' Identifier '{' domain_attribute_list '}'
= 'DOMAIN' Identifier '{' '}'

domain_attribute_list
= domain_attribute
= domain_attribute_list domain_attribute

domain_attribute
= handling /* O */
= response_codes /* O */

```

A.6.10 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
= display_items /* O */
= edit_items /* M */
= label /* M */
= post_edit_actions /* O */
= pre_edit_actions /* O */

edit_items
= 'EDIT_ITEMS' '{' edit_items_specifier_list '}'

edit_items_specifier_list
= edit_items_specifier

```

```

    = edit_items_specifier_list edit_items_specifier

edit_items_specifier
    = edit_item_list
    = 'IF' '(' expr ')' '{' edit_items_specifier_list '}'
    = 'IF' '(' expr ')' '{' edit_items_specifier_list '}'
    'ELSE' '{' edit_items_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' edit_items_selection_list '}'

edit_items_selection_list
    = edit_items_selection
    = edit_items_selection_list edit_items_selection

edit_items_selection
    = 'CASE' expr ':' edit_items_specifier_list
    = 'DEFAULT' ':' edit_items_specifier_list

edit_item_list
    = edit_item
    = edit_item_list ',' edit_item

edit_item
    = reference_without_call

display_items
    = 'DISPLAY_ITEMS' '{' display_items_specifier_list '}'

display_items_specifier_list
    = display_items_specifier
    = display_items_specifier_list display_items_specifier

display_items_specifier
    = display_item_list
    = 'IF' '(' expr ')' '{' display_items_specifier_list '}'
    = 'IF' '(' expr ')' '{' display_items_specifier_list '}'
    'ELSE' '{' display_items_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' display_items_selection_list '}'

display_items_selection_list
    = display_items_selection
    = display_items_selection_list display_items_selection

display_items_selection
    = 'CASE' expr ':' display_items_specifier_list
    = 'DEFAULT' ':' display_items_specifier_list

display_item_list
    = display_item
    = display_item_list ',' display_item

display_item
    = variable_reference

```

A.6.11 FILE

```

file
    = 'FILE' Identifier '{' file_attribute_list '}'

file_attribute_list
    = file_attribute_listR

file_attribute_listR
    = file_attribute
    = file_attribute_listR file_attribute

file_attribute
    = file_label

```

/* 0 */


```

= file_help          /* O */
= file_members       /* M */

file_label
= label

file_help
= help

file_members
= members

```

A.6.12 GRAPH

```

graph
= 'GRAPH' Identifier '{' graph_attribute_list '}'

graph_attribute_list
= graph_attribute_listR

graph_attribute_listR
= graph_attribute
= graph_attribute_listR graph_attribute

graph_attribute
= graph_label
= graph_help
= graph_members
= graph_x_axis
= graph_height
= graph_width
= graph_cycle_time

graph_label
= label

graph_help
= help

graph_members
= members

graph_x_axis
= 'X_AXIS' axis_specifier

graph_height
= 'HEIGHT' display_size_specifier

graph_width
= 'WIDTH' display_size_specifier

display_size_specifier
= display_size_keyword ';'
= 'IF' '(' expr ')' '{' display_size_specifier '}'
= 'IF' '(' expr ')' '{' stmt1: display_size_specifier '}'
  'ELSE' '{' stmt2: display_size_specifier '}'
= 'SELECT' '(' expr ')' '{' display_size_selection_list '}'

display_size_selection_list
= display_size_selection_listR

display_size_selection_listR
= display_size_selection
= display_size_selection_listR display_size_selection

display_size_selection
= 'CASE' expr ':' display_size_specifier

```

```

    = 'DEFAULT' ':' display_size_specifier

display_size_keyword
    = 'XX_SMALL'
    = 'X_SMALL'
    = 'SMALL'
    = 'MEDIUM'
    = 'XX_LARGE'
    = 'X_LARGE'
    = 'LARGE'

graph_cycle_time
    = 'CYCLE_TIME' expr_specifier

```

A.6.13 GRID

```

grid
    = 'GRID' Identifier '{' grid_attribute_list '}'

grid_attribute_list
    = grid_attribute_listR

grid_attribute_listR
    = grid_attribute
    = grid_attribute_listR grid_attribute

grid_attribute
    = grid_handling
    = grid_height
    = grid_help
    = grid_label
    = grid_orientation
    = grid_validity
    = grid_vectors
    = grid_width

grid_handling
    = 'HANDLING' handling_specifier

grid_height
    = 'HEIGHT' display_size_specifier

grid_help
    = help

grid_label
    = label

grid_orientation
    = 'ORIENTATION' orientation_specifier ';'

orientation_specifier
    = 'HORIZONTAL'
    = 'VERTICAL'

grid_vectors
    = 'VECTORS' '{' vector_specifier_list '}'

vector_specifier_list
    = vector_specifier_listR

vector_specifier_listR
    = vector_specifier
    = vector_specifier_listR ',' vector_specifier

vector_specifier
    = vector

```

```

= 'IF' '(' expr ')' '{' vector_specifier_list '}'
= 'IF' '(' expr ')' '{' stmt1: vector_specifier_list '}'
  'ELSE' '{' stmt2: 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_listR

vector_values_listR
= vector_values
= vector_values_listR ',' vector_values

vector_values
= reference_without_call
= Integer
= Real
= string

grid_width
= 'WIDTH' display_size_specifier

```

A.6.14 IMAGE

```

image
= 'IMAGE' Identifier '{' image_attribute_list '}'

image_attribute_list
= image_attribute_listR

image_attribute_listR
= image_attribute
= image_attribute_listR image_attribute

image_attribute
= image_label /* O */
= image_help /* O */
= image_link /* O */
= image_path /* M */
= image_validity /* O */

image_label
= label

image_help
= help

image_link
= 'LINK' link_specifier

link_specifier
= reference_no_call ';'
= 'IF' '(' expr ')' '{' link_specifier '}'
= 'IF' '(' expr ')' '{' stmt1: link_specifier '}'
  'ELSE' '{' stmt2: link_specifier '}'
= 'SELECT' '(' expr ')' '{' link_selection_list '}'

```

```

link_selection_list
    = link_selection_listR

link_selection_listR
    = link_selection
    = link_selection_listR link_selection

link_selection
    = 'CASE' expr ':' link_specifier
    = 'DEFAULT' ':' link_specifier

image_path
    = 'PATH' string_specifier

image_validity
    = validity

```

A.6.15 LIST

```

list
    = 'LIST' Identifier '{' list_attribute_list '}'

list_attribute_list
    = list_attribute_listR

list_attribute_listR
    = list_attribute
    = list_attribute_listR list_attribute

list_attribute
    = list_label /* O */
    = list_help /* O */
    = list_capacity /* O */
    = list_count /* O */
    = list_type /* M */

list_label
    = label

list_help
    = help

list_capacity
    = 'CAPACITY' expr_specifier ';'

list_count
    = 'COUNT' expr_specifier ';'

list_type
    = 'TYPE' list_type_specifier ';'

list_type_specifier
    = reference_without_call

```

A.6.16 IMPORT

```

imported_description
    = 'IMPORT' identification '{' imports '}'
    = 'IMPORT' identification '{' imports redefinitions '}'
    = 'IMPORT' '"' filename '"'
    = 'IMPORT' '<' filename '>'

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'
    = 'CONNECTIONS'
    = 'DOMAINS'
    = 'EDIT_DISPLAYS'
    = 'FILES'
    = 'GRAPHS'
    = 'ITEM_ARRAYS'
    = 'LISTS'
    = 'MENUS'
    = 'METHODS'
    = 'PROGRAMS'
    = 'RECORDS'
    = 'REFRESHES'
    = 'RELATIONS'
    = 'RESPONSE_CODES'
    = 'SOURCES'
    = 'UNITS'
    = 'VARIABLES'
    = 'VARIABLE_LISTS'
    = 'WRITE_AS_ONES'
    = 'WAVEFORMS'

redefinitions
    = 'REDEFINITIONS' '{' redefinition_list '}'

redefinition_list
    = redefinition
    = redefinition_list redefinition

redefinition
    = array_redefinition
    = axis_redefinition
    = block_redefinition
    = chart_redefinition
    = collection_redefinition
    = command_redefinition
    = connection_redefinition
    = domain_redefinition
    = edit_display_redefinition
    = file_definition
    = graph_definition
    = item_array_redefinition
    = list_redefinition
    = menu_redefinition
    = method_redefinition
    = program_redefinition
    = record_redefinition

```

```

= refresh_relation_redefinition
= response_codes_definition_redefinition
= source_redefinition
= unit_relation_redefinition
= variable_redefinition
= variable_list_redefinition
= wao_relation_redefinition
= waveform_redefinition

```

A.6.17 LIKE

like

```

= Identifier 'LIKE' Identifier
  '{' attribute_redefinition_list '}'
= Identifier 'LIKE' 'ARRAY' Identifier
  '{' array_attribute_redefinition_list '}'
= Identifier 'LIKE' 'BLOCK' Identifier
  '{' block_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COLLECTION' 'OF' item_type Identifier
  '{' collection_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COMMAND' Identifier
  '{' command_attribute_redefinition_list '}'
= Identifier 'LIKE' 'CONNECTION' Identifier
  '{' connection_attribute_redefinition_list '}'
= Identifier 'LIKE' 'DOMAIN' Identifier
  '{' domain_attribute_redefinition_list '}'
= Identifier 'LIKE' 'EDIT_DISPLAY' Identifier
  '{' edit_display_attribute_redefinition_list '}'
= Identifier 'LIKE' 'ITEM_ARRAY' 'OF' item_type
  Identifier
  '{' item_array_attribute_redefinition_list '}'
= Identifier 'LIKE' 'MENU' Identifier
  '{' menu_attribute_redefinition_list '}'
= Identifier 'LIKE' 'METHOD' Identifier
  '{' method_attribute_redefinition_list '}'
= Identifier 'LIKE' 'PROGRAM' Identifier
  '{' program_attribute_redefinition_list '}'
= Identifier 'LIKE' 'RECORD' Identifier
  '{' record_attribute_redefinition_list '}'
= Identifier 'LIKE' 'REFRESH' Identifier
  '{' '}'
= Identifier 'LIKE' 'RESPONSE_CODES' Identifier
  '{' response_code_redefinition_list '}'
= Identifier 'LIKE' 'UNIT' Identifier
  '{' '}'
= Identifier 'LIKE' 'VARIABLE' Identifier
  '{' variable_attribute_redefinition_list '}'
= Identifier 'LIKE' 'VARIABLE_LIST' Identifier
  '{' variable_list_attribute_redefinition_list '}'
= Identifier 'LIKE' 'WRITE_AS_ONE' Identifier
  '{' '}'
= Identifier 'LIKE' 'AXIS' Identifier
  '{' '}'
= Identifier 'LIKE' 'CHART' Identifier
  '{' '}'
= Identifier 'LIKE' 'GRAPH' Identifier
  '{' '}'
= Identifier 'LIKE' 'FILE' Identifier
  '{' '}'
= Identifier 'LIKE' 'LIST' Identifier
  '{' '}'
= Identifier 'LIKE' 'SOURCE' Identifier
  '{' '}'
= Identifier 'LIKE' 'WAVEFORM' Identifier
  '{' '}'

```

```

attribute_redefinition_list
= attribute_redefinition

```

```

= attribute_redefinition_list attribute_redefinition

attribute_redefinition
= appinstance_redefinition
= arguments_redefinition
= array_type_redefinition
= block_a_characteristics_redefinition
= block_a_collection_item_redefinition
= block_a_edit_disp_item_redefinition
= block_a_item_array_item_redefinition
= block_a_menu_item_redefinition
= block_a_method_item_redefinition
= block_a_parameters_redefinition
= block_a_parameter_lists_redefinition
= block_a_refresh_item_redefinition
= block_a_unit_item_redefinition
= block_a_wao_item_redefinition
= block_b_number_redefinition
= block_b_type_redefinition
= command_slot_redefinition
= command_index_redefinition
= command_block_redefinition
= command_operation_redefinition
= command_transaction_redefinition
= command_connection_redefinition
= command_module_redefinition
= display_items_redefinition
= edit_items_redefinition
= elements_redefinition
= handling_redefinition
= help_redefinition
= members_redefinition
= menu_access_redefinition
= menu_entry_redefinition
= menu_items_redefinition
= menu_purpose_redefinition
= menu_style_redefinition
= method_definition_redefinition
= number_of_elements_redefinition
= optional_label_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
= read_timeout_redefinition
= response_code_redefinition
= response_codes_reference_redefinition
= type_redefinition
= validity_redefinition
= variable_class_redefinition
= write_timeout_redefinition

```

A.6.18 MENU

```

menu
= 'MENU' Identifier '{' menu_attribute_list '}'

menu_attribute_list
= menu_attribute
= menu_attribute_list menu_attribute

menu_attribute
= help /* O */
= label /* M */
= menu_access /* O */

```

```

= menu_entry          /* O */
= menu_items          /* M */
= menu_purpose          /* O */
= menu_role           /* O */
= menu_style          /* O */
= validity            /* O */
= pre_edit_actions    /* O */
= pre_read_actions    /* O */
= pre_write_actions   /* O */
= post_edit_actions   /* O */
= post_read_actions   /* O */
= post_write_actions  /* O */

menu_access
= 'ACCESS' 'ONLINE' ';'
= 'ACCESS' 'OFFLINE' ';'

menu_entry
= 'ENTRY' menu_entry_specifier

menu_entry_specifier
= menu_entry_item ';'
= 'IF' '(' expr ')' '{' menu_entry_specifier '}'
= 'IF' '(' expr ')' '{' menu_entry_specifier '}'
  'ELSE' '{' menu_entry_specifier '}'
= 'SELECT' '(' expr ')' '{' menu_entry_selection_list '}'

menu_entry_selection_list
= menu_entry_selection
= menu_entry_selection_list menu_entry_selection

menu_entry_selection
= 'CASE' expr ':' menu_entry_specifier
= 'DEFAULT' ':' menu_entry_specifier

menu_entry_item
= 'ROOT'

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_list

menu_item_selection_list
= menu_item_selection
= menu_item_selection_list menu_item_selection

menu_item_selection
= 'CASE' expr ':' menu_item_specifier_list
= 'DEFAULT' ':' menu_item_specifier_list

menu_item_list
= menu_item
= menu_item_list ',' menu_item

```



```

menu_item
= reference_without_call
= reference_without_call menu_item_args
= 'SEPARATOR'
= 'COLUMNBREAK'
= 'ROWBREAK'

menu_item_args
= '(' argument_list ')'
= '(' 'REVIEW' ')'
= '(' variable_qualifier_list ')'
= '(' image_qualifier_list ')'

variable_qualifier_list
= variable_qualifier
= variable_qualifier_list ',' variable_qualifier

variable_qualifier
= 'DISPLAY_VALUE'
= 'READ_ONLY'
= 'HIDDEN'
= 'NO_LABEL'
= 'NO_UNIT'

image_qualifier_list
= image_qualifier
= image_qualifier_list ',' image_qualifier

image_qualifier
= 'INLINE'

menu_purpose
= 'PURPOSE' menu_purpose_specifier

menu_purpose_specifier
= menu_purpose_list ';'
= 'IF' '(' expr ')' '{' menu_purpose_specifier '}'
= 'IF' '(' expr ')' '{' menu_purpose_specifier '}'
= 'ELSE' '{' menu_purpose_specifier '}'
= 'SELECT' '(' expr ')' '{' menu_purpose_selection_list '}'

menu_purpose_selection_list
= menu_purpose_selection
= menu_purpose_selection_list menu_purpose_selection

menu_purpose_selection
= 'CASE' expr ':' menu_purpose_specifier
= 'DEFAULT' ':' menu_purpose_specifier

menu_purpose_list
= menu_purpose_item
= menu_purpose_list '&' menu_purpose_item

menu_purpose_item
= 'CATALOG'
= 'DIAGNOSE'
= 'LOAD_TO_APPLICATION'
= 'LOAD_TO_DEVICE'
= 'MENU'
= 'PROCESS_VALUE'
= 'SIMULATION'
= 'TABLE'
= string

menu_role
= 'ROLE' menu_role_specifier

```

```

menu_role_specifier
= menu_role_list ';'
= 'IF' '(' expr ')' '{' menu_role_specifier '}'
= 'IF' '(' expr ')' '{' menu_role_specifier '}'
'ELSE' '{' menu_role_specifier '}'
= 'SELECT' '(' expr ')' '{' menu_role_selection_list '}'

menu_role_selection_list
= menu_role_selection
= menu_role_selection_list menu_role_selection

menu_role_selection
= 'CASE' expr ':' menu_role_specifier
= 'DEFAULT' ':' menu_role_specifier

menu_role_list
= menu_role_item
= menu_role_list '&' menu_role_item

menu_role_item
= string

menu_style
= 'STYLE' menu_style_specifier

menu_style_specifier
= menu_style_item ';'
= 'IF' '(' expr ')' '{' menu_style_specifier '}'
= 'IF' '(' expr ')' '{' menu_style_specifier '}'
'ELSE' '{' menu_style_specifier '}'
= 'SELECT' '(' expr ')' '{' menu_style_selection_list '}'

menu_style_selection_list
= menu_style_selection
= menu_style_selection_list menu_style_selection

menu_style_selection
= 'CASE' expr ':' menu_style_specifier
= 'DEFAULT' ':' menu_style_specifier

menu_style_item = 'DIALOG'
= 'GROUP'
= 'MENU'
= 'PAGE'
= 'TABLE'
= 'WINDOW'
= string

```

A.6.19 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

```

```

    = 'char'
    = 'unsigned char'
= 'int'
= 'unsigned int'
    = 'long'
    = 'unsigned long'
= 'float'
    = 'double'
    = 'DD_STRING'
    = 'DD_ITEM'

method_attribute_list
    = method_attribute
    = method_attribute_list method_attribute

method_attribute
    = help
    = label
    = method_access
    = method_definition
    = method_type
    = validity
    = variable_class

method_access
    = 'ACCESS' 'OFFLINE' ';'
    = 'ACCESS' 'ONLINE' ';'

method_definition
    = 'DEFINITION' c_compound_statement

method_type
    = 'TYPE' method_return_type

method_return_type
    = 'char'
    = 'unsigned char'
= 'int'
= 'unsigned int'
    = 'long'
    = 'unsigned long'
= 'float'
    = 'double'
    = 'DD_STRING'

A.6.20 PROGRAM

program
    = 'PROGRAM' Identifier '{' program_attribute_list '}'
    = 'PROGRAM' Identifier '{' '}'

program_attribute_list
    = program_attribute
    = program_attribute_list program_attribute

program_attribute
    = arguments
    = response_codes

arguments
    = 'ARGUMENTS' '{' '}'
    = 'ARGUMENTS' '{' arguments_specifier_list '}'

arguments_specifier_list
    = arguments_specifier
    = arguments_specifier_list arguments_specifier

```

```

arguments_specifier
= argument_list
= 'IF' '(' expr ')' '{' arguments_specifier_list '}'
= 'IF' '(' expr ')' '{' arguments_specifier_list '}'
  'ELSE' '{' arguments_specifier_list '}'
= 'SELECT' '(' expr ')' '{' arguments_selection_list '}'

arguments_selection_list
= arguments_selection
= arguments_selection_list arguments_selection

arguments_selection
= 'CASE' expr ':' arguments_specifier_list
= 'DEFAULT' ':' arguments_specifier_list

argument_list
= argument
= argument_list ',' argument

argument
= Integer
= variable_reference

```

A.6.21 RECORDS

```

record
= 'RECORD' Identifier '{' record_attribute_list '}'

record_attribute_list
= record_attribute
= record_attribute_list record_attribute

record_attribute
= help /* O */
= label /* M */
= members /* M */
= response_codes /* O */

```

A.6.22 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.

NOTE If an EDD application supports more than one profile, the keyword ARRAY may be also used instead of ITEM_ARRAY. In this case the semantical selection is done by attributes.

```

item_array
= 'ITEM_ARRAY' 'OF' item_array_attribute_type Identifier '{'
item_array_attribute_list '}'
= 'ARRAY' 'OF' item_array_attribute_type Identifier '{'
item_array_attribute_list '}'

item_array_attribute_type
= item_array_type

item_array_type
= 'ARRAY'
= 'BLOCK'
= 'COLLECTION' 'OF' item_array_type
= 'COMMAND'
= 'CONNECTION'
= 'DOMAIN'
= 'ITEM_ARRAY' 'OF' item_array_type
= 'MENU'
= 'METHOD'
= 'PROGRAM'

```

```

= 'RECORD'
= 'REFRESH'
= 'RESPONSE_CODES'
= 'UNIT'
= 'VARIABLE'
= 'VARIABLE_LIST'
= 'WRITE_AS_ONE'

item_array_attribute_list
= item_array_attribute
= item_array_attribute_list item_array_attribute

item_array_attribute
= elements /* M */
= help /* O */
= label /* O */

elements
= 'ELEMENTS' '{' elements_specifier_list '}'

elements_specifier_list
= elements_specifier
= elements_specifier_list elements_specifier

elements_specifier
= element
= 'IF' '(' expr ')' '{' elements_specifier_list '}'
= 'IF' '(' expr ')' '{' elements_specifier_list '}'
= 'ELSE' '{' elements_specifier_list '}'
= 'SELECT' '(' expr ')' '{' elements_selection_list '}'

elements_selection_list
= elements_selection
= elements_selection_list elements_selection

elements_selection
= 'CASE' expr ':' elements_specifier_list
= 'DEFAULT' ':' elements_specifier_list

element
= Integer ',' reference_without_call ';'
= Integer ',' reference_without_call ',' description_string ';'
= Integer ',' reference_without_call ',' description_string ','
help_string ';'

```

A.6.23 Relations

```

refresh_relation
= 'REFRESH' Identifier
  '{' refresh_relation_left ':' refresh_relation_right '}'

refresh_relation_left
= variable_reference_specifier_list

refresh_relation_right
= variable_reference_specifier_list

unit_relation
= 'UNIT' Identifier
  '{' unit_relation_left ':' unit_relation_right '}'

unit_relation_left
= variable_reference_specifier

unit_relation_right
= variable_axis_reference_specifier_list

```

```

wao_relation
= 'WRITE_AS_ONE' Identifier '{' wao_specifier '}'

wao_specifier
= variable_reference_specifier_list

variable_reference_specifier_list
= variable_reference_specifier
= variable_reference_specifier_list variable_reference_specifier
= variable_reference_specifier_list ',' variable_reference_specifier

variable_reference_specifier
= variable_reference
= 'IF' '(' expr ')' '{' variable_reference_specifier '}'
= 'IF' '(' expr ')' '{' variable_reference_specifier '}'
'ELSE' '{' variable_reference_specifier '}'
= 'SELECT' '(' expr ')' '{' variable_reference_selection_list '}'

variable_reference_selection_list
= variable_reference_selection
= variable_reference_selection_list variable_reference_selection

variable_reference_selection
= 'CASE' expr ':' variable_reference_specifier
= 'DEFAULT' ':' variable_reference_specifier

variable_axis_reference_selection_list
= variable_axis_reference_selection_listR

variable_axis_reference_selection_listR
= variable_axis_reference_selection
= variable_axis_reference_selection_listR

variable_axis_reference_selection
= 'CASE' expr ':' variable_axis_reference_specifier
= 'DEFAULT' ':' variable_axis_reference_specifier

variable_axis_reference_specifier
= variable_reference
= axis_reference
= 'IF' '(' expr ')' '{' variable_axis_reference_specifier '}'
= 'IF' '(' expr ')' '{' stmt1: variable_axis_reference_specifier '}'
'ELSE' '{' stmt2: variable_axis_reference_specifier '}'
= 'SELECT' '(' expr ')' '{' variable_axis_reference_selection_list '}'

```

A.6.24 RESPONSE_CODES

```

response_codes_definition
= 'RESPONSE_CODES' Identifier '{' response_codes_attribute_list
'}'

response_codes_attribute_list
= response_codes_specifier_list

response_codes_specifier_list
= response_codes_specifier
= response_codes_specifier_list response_codes_specifier

response_codes_specifier
= response_code
= 'IF' '(' expr ')' '{' response_codes_specifier_list '}'
= 'IF' '(' expr ')' '{' response_codes_specifier_list '}'
'ELSE' '{' response_codes_specifier_list '}'
= 'SELECT' '(' expr ')' '{' response_codes_selection_list '}'

response_codes_selection_list

```

```

= response_codes_selection
= response_codes_selection_list response_codes_selection

response_codes_selection
= 'CASE' expr ':' response_codes_specifier_list
= 'DEFAULT' ':' response_codes_specifier_list

response_code
= Integer ',' response_code_type ',' description_string ';'
= Integer ',' response_code_type ',' description_string ','
  help_string ';'

response_code_type
= 'SUCCESS'
= 'MISC_WARNING'
= 'DATA_ENTRY_WARNING'
= 'DATA_ENTRY_ERROR'
= 'MODE_ERROR'
= 'PROCESS_ERROR'
= 'MISC_ERROR'

```

A.6.25 SOURCE

```

source
= 'SOURCE' Identifier '{' source_attribute_list '}'

source_attribute_list
= source_attribute_listR

source_attribute_listR
= source_attribute
= source_attribute_listR source_attribute

source_attribute
= source_help /* O */
= source_label /* O */
= source_emphasis /* O */
= source_line_color /* O */
= source_line_type /* O */
= source_y_axis /* O */
= source_members /* M */
= source_init_actions /* O */
= source_refresh_actions /* O */
= source_exit_actions /* O */

source_help
= help

source_label
= label

source_emphasis
= 'EMPHASIS' boolean_specifier

source_line_color
= 'LINE_COLOR' line_color_specifier

source_line_type
= 'LINE_TYPE' line_type_specifier ';'

source_y_axis
= 'Y_AXIS' axis_specifier

source_members
= members

source_init_actions

```

```

    = 'INIT_ACTIONS' '{' actions_specifier_list '}'

source_refresh_actions
    = 'REFRESH_ACTIONS' '{' actions_specifier_list '}'

source_exit_actions
    = 'EXIT_ACTIONS' '{' actions_specifier_list '}'

```

A.6.26 VALUE_ARRAY

```

array
    = 'ARRAY' Identifier '{' array_attribute_list '}'

array_attribute_list
    = array_attribute
    = array_attribute_list array_attribute

array_attribute
    = array_type                /* M */
    = array_size                /* M */
    = help                      /* O */
    = label                    /* M */
    = response_codes            /* O */

array_type
    = 'TYPE' Identifier ';'

array_size
    = 'NUMBER_OF_ELEMENTS' Integer ';'

```

A.6.27 VARIABLE

```

variable
    = 'VARIABLE' Identifier '{' variable_attribute_list '}'

variable_attribute_list
    = variable_attribute
    = variable_attribute_list variable_attribute

variable_attribute
    = constant_unit            /* O */
    = default_value            /* O */
    = handling                 /* O */
    = help                     /* O */
    = initial_value            /* O */
    = label                    /* M */
    = post_edit_actions        /* O */
    = post_read_actions        /* O */
    = post_write_actions       /* O */
    = pre_edit_actions         /* O */
    = pre_read_actions         /* O */
    = pre_write_actions        /* O */
    = refresh_actions          /* O */
    = read_timeout             /* O */
    = response_codes           /* O */
    = type                     /* M */
    = validity                 /* O */
    = variable_class           /* M */
    = variable_style           /* O */
    = write_timeout            /* 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'
    = 'CONTAINED'
    = 'CORRECTION'
    = 'DEVICE'
    = 'DIAGNOSTIC'
    = 'DIGITAL_INPUT'
    = 'DIGITAL_OUTPUT'
    = 'DISCRETE_INPUT'
    = 'DISCRETE_OUTPUT'
    = 'DYNAMIC'
    = 'FREQUENCY_INPUT'
    = 'FREQUENCY_OUTPUT'
    = 'HART'
    = 'INPUT'
    = 'LOCAL'
    = 'LOCAL_DISPLAY'
    = 'OPERATE'
    = 'OUTPUT'
    = 'SERVICE'
    = 'TUNE'

variable_style
    = 'STYLE' string ';'

constant_unit
    = 'CONSTANT_UNIT' string_specifier

default_value
    = 'DEFAULT_VALUE' expr_specifier

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_definition_

handling_definition_
    = handling_keyword
    = handling_definition_ '&' handling_keyword

handling_keyword
    = 'READ'
    = 'WRITE'

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_list
    = '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_list
    = method_specification
    = method_list ',' method_specification

method_specification
    = method_reference
    = method_definition

read_timeout
    = 'READ_TIMEOUT' expr_specifier

write_timeout
    = 'WRITE_TIMEOUT' expr_specifier

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

```

```

type
    = 'TYPE' type_specifier

type_specifier
    = arithmetic_type
    = enumerated_type
    = index_type
    = string_type
    = date_time_type
    = object_reference

arithmetic_type
    = 'INTEGER' arithmetic_option_size arithmetic_options
    = 'UNSIGNED_INTEGER' arithmetic_option_size arithmetic_options
    = 'DOUBLE' arithmetic_options
    = 'FLOAT' arithmetic_options

arithmetic_option_size
    =
    = '(' Integer ')'

arithmetic_options
    = ';'
    = '{' arithmetic_option_list '}'

arithmetic_option_list
    = arithmetic_option
    = arithmetic_option_list arithmetic_option

arithmetic_option
    = arithmetic_default_value
    = arithmetic_initial_value
    = display_format
    = edit_format
    = enumerator_list
    = maximum_value
    = maximum_value_n
    = minimum_value
    = minimum_value_n
    = scaling_factor

arithmetic_default_value
    = 'DEFAULT_VALUE' expr_specifier

arithmetic_initial_value
    = 'INITIAL_VALUE' expr_specifier

display_format
    = 'DISPLAY_FORMAT' string_specifier

edit_format
    = 'EDIT_FORMAT' string_specifier

scaling_factor
    = 'SCALING_FACTOR' expr_specifier

maximum_value
    = 'MAX_VALUE' expr_specifier

maximum_value_n
    = MAX_VALUE_n expr_specifier

minimum_value
    = 'MIN_VALUE' expr_specifier

minimum_value_n

```

```

= MIN_VALUE_n expr_specifier

enumerated_type
= 'ENUMERATED' enumerated_option_size '{' enumerated_option_list
  '}'
= 'BIT_ENUMERATED' enumerated_option_size '{'
  bit_enumerated_option_list '}'

enumerated_option_size
=
= '(' Integer ')'

enumerated_option_list
= enumerated_option
= enumerated_option_list enumerated_option

enumerated_option
= arithmetic_default_value
= arithmetic_initial_value
= enumerators_specifier

enumerators_specifier_list
= enumerators_specifier
= enumerators_specifier_list enumerators_specifier

enumerators_specifier
= 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

enumerator_list
= enumerator
= enumerator_list ',' enumerator

enumerator
= '{' enumerator_value ',' description_string '}'
= '{' enumerator_value ',' description_string ',' help_string '}'

enumerator_value
= Integer
= Real
= string

description_string
= string

help_string
= string

bit_enumerated_option_list
= bit_enumerated_option
= bit_enumerated_option_list bit_enumerated_option

bit_enumerated_option
= arithmetic_default_value
= arithmetic_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 ',' description_string
  '}'
= '{' Integer ',' description_string ','
  help_string
  '}'
= '{' Integer ',' description_string ','
  variable_class_definition_wrap
  '}'
= '{' Integer ',' description_string ','
  status_class
  '}'
= '{' Integer ',' description_string ','
  help_string ',' variable_class_definition_wrap
  '}'
= '{' Integer ',' description_string ','
  help_string ',' status_class
  '}'
= '{' Integer ',' description_string ','
  help_string ',' method_reference
  '}'
= '{' Integer ',' description_string ','
  variable_class_definition_wrap ',' status_class
  '}'
= '{' Integer ',' description_string ','
  variable_class_definition_wrap ',' method_reference
  '}'
= '{' Integer ',' description_string ','
  status_class ',' method_reference
  '}'
= '{' Integer ',' description_string ','
  help_string ',' variable_class_definition_wrap ',' status_class
  '}'
= '{' Integer ',' description_string ','
  help_string ',' variable_class_definition_wrap ',' method_reference
  '}'
= '{' Integer ',' description_string ','
  help_string ',' status_class ',' method_reference
  '}'
= '{' Integer ',' description_string ','
  variable_class_definition_wrap ',' status_class ',' method_reference
  '}'
= '{' Integer ',' description_string ','

```

```

        help_string ',' variable_class_definition_wrap ',' status_class ','
method_reference
    '}'

variable_class_definition_wrap
    = variable_class_definition

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'
    = 'IGNORE_IN_TEMPORARY_MASTER'
    = 'BAD'

index_type
    = 'INDEX' item_array_reference ';'
    = 'INDEX' '(' Integer ')' item_array_reference ';'

string_type
    = string_option_type string_option_size ';'
    = string_option_type string_option_size '{' '}'
    = string_option_type string_option_size '{' string_option_list '}'

string_option_type
    = string_keyword

string_keyword
    = 'ASCII'
    = 'BITSTRING'
    = 'EUC'
    = 'PACKED_ASCII'
    = 'PASSWORD'
    = 'VISIBLE'
    = 'OCTET'

string_option_size
    = '(' Integer ')'

string_option_list
    = string_option
    = string_option_list string_option

string_option
    = string_default_value
    = string_initial_value
    = enumerator_list

string_default_value

```

```

    = 'DEFAULT_VALUE' string ';'

string_initial_value
    = 'INITIAL_VALUE' string ';'

date_time_type
    = 'DATE' ';'
    = 'DATE' '{' '}'
    = 'DATE' '{' string_option_list '}'
    = 'DATE_AND_TIME' ';'
    = 'DATE_AND_TIME' '{' '}'
    = 'DATE_AND_TIME' '{' string_option_list '}'
    = 'DURATION' ';'
    = 'DURATION' '{' '}'
    = 'DURATION' '{' string_option_list '}'
    = 'TIME' ';'
    = 'TIME' '{' '}'
    = 'TIME' '{' string_option_list '}'
    = 'TIME' '(' Integer ')' ';'
    = 'TIME' '(' Integer ')' '{' '}'
    = 'TIME' '(' Integer ')' '{' string_option_list '}'
    = 'TIME_VALUE' ';'
    = 'TIME_VALUE' '{' '}'
    = 'TIME_VALUE' '{' string_option_list '}'

```

```

object_reference
    = 'OBJECT_REFERENCE' object_reference_value ';'

```

```

object_reference_value
    = 'ROOT'
    = 'PARENT'
    = 'CHILD'
    = 'SELF'
    = 'NEXT'
    = 'PREV'
    = 'FIRST'
    = 'LAST'

```

A.6.28 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
    = help                /* O */
    = label               /* O */
    = members             /* M */
    = response_codes      /* O */

```

A.6.29 WAVEFORM

```

waveform
    = 'WAVEFORM' Identifier '{' waveform_attribute_list '}'

waveform_attribute_list
    = waveform_attribute_listR

waveform_attribute_listR
    = waveform_attribute
    = waveform_attribute_listR waveform_attribute

waveform_attribute
    = waveform_help
    = waveform_label
    = waveform_type
    = waveform_handling
    = waveform_emphasis
    = waveform_exit_actions
    = waveform_line_color
    = waveform_line_type
    = waveform_y_axis
    = waveform_init_actions
    = waveform_refresh_actions
    = waveform_key_points

    /* O */
    /* O */
    /* M */
    /* O */
    /* O */
    /* O */
    /* O */
    /* O */
    /* O */
    /* O */
    /* O */

waveform_help
    = help

waveform_label
    = label

waveform_handling
    = handling

waveform_emphasis
    = 'EMPHASIS' boolean_specifier

waveform_exit_actions
    = 'EXIT_ACTIONS' '{' actions_specifier_list '}'

waveform_line_color
    = 'LINE_COLOR' line_color_specifier

waveform_line_type
    = 'LINE_TYPE' line_type_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_listR

line_color_selection_listR
    =
    = line_color_selection_listR line_color_selection line_color_selection

line_color_selection
    = 'CASE' expr ':' line_color_specifier
    = 'DEFAULT' ':' line_color_specifier

color
    = Integer

```



```

    = variable_reference

line_type_specifier
    = line_type_keyword ';'
    = 'IF' '(' expr ')' '{' line_type_specifier '}'
    = 'IF' '(' expr ')' '{' stmt1: line_type_specifier '}'
    'ELSE' '{' stmt2: line_type_specifier '}'
    = 'SELECT' '(' expr ')' '{' line_type_selection_list '}'

line_type_selection_list
    = line_type_selection_listR

line_type_selection_listR
    = line_type_selection
    = line_type_selection_listR line_type_selection

line_type_selection
    = 'CASE' expr ':' line_type_specifier
    = 'DEFAULT' ':' line_type_specifier

line_type_keyword
    = 'DATA'
    = 'DATA' digit
    = 'LOW_LOW_LIMIT'
    = 'LOW_LIMIT'
    = 'HIGH_LIMIT'
    = 'HIGH_HIGH_LIMIT'
    = 'TRANSPARENT'

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_listR

waveform_yt_type_option_listR
    = waveform_yt_type_option
    = waveform_yt_type_option_listR 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_listR

values_option_listR
    = values_option
    = values_option_listR ',' 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_listR

waveform_xy_type_option_listR
    = waveform_xy_type_option
    = waveform_xy_type_option_listR 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 ')' '{' stmt1: axis_specifier '}'
    'ELSE' '{' stmt2: axis_specifier '}'
    = 'SELECT' '(' expr ')' '{' axis_selection_list '}'

axis_selection_list
    = axis_selection_listR

axis_selection_listR
    = axis_selection
    = axis_selection_listR axis_selection

axis_selection
    = 'CASE' expr ':' axis_specifier
    = 'DEFAULT' ':' axis_specifier

waveform_init_actions
    = 'INIT_ACTIONS' '{' actions_specifier_list '}'

waveform_refresh_actions
    = 'REFRESH_ACTIONS' '{' actions_specifier_list '}'

waveform_key_points
    = 'KEY_POINTS' '{' key_points_options_list '}'

key_points_options_list
    = key_points_options_listR

```

```
key_points_options_listR
    = key_points_options
    = key_points_options_listR key_points_options
```

```
key_points_options
    = waveform_x_values          /* M */
    = waveform_y_values          /* M */
```

A.6.30 Common attributes

```
help
    = 'HELP' string_specifier
```

```
label
    = 'LABEL' string_specifier
```

```
members
    = 'MEMBERS' '{' members_specifier_list '}'
```

```
number_specifier:
    = number_items ';'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    = 'ELSE' '{' number_specifier '}'
    = 'SELECT' '(' expr ')' '{' number_selection_list '}'
```

```
number_items:
    = expr
    = variable_reference
```

```
response_codes
    = 'RESPONSE_CODES' '{' response_codes_specifier_list '}'
    = 'RESPONSE_CODES' response_code_reference_specifier
```

```
validity
    = 'VALIDITY' boolean_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_literal
    = variable_reference
    = variable_reference '(' Integer ')'
    = dictionary_reference
```

```
string_literal
    = String
    = string_literal String
```

```

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 ',' description_string ';'
    = Identifier ',' reference_without_call ',' description_string ','
help_string
    ';'

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
    = 'CASE' expr ':' response_code_reference_specifier
    = 'DEFAULT' ':' response_code_reference_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'

```

A.6.31 OPEN, CLOSE

```

open
    = 'OPEN' filename

```

```
close
    = 'CLOSE' filename
```

```
filename
    = Identifier
```

A.6.32 Expression

```
primary_expr
    = reference
    = conditional_expr
    = String
    = 'PI'
    = '(' expr ') '
    = primary_expr_keyword
```

```
primary_expr_keyword
    = axis_scaling_keyword
```

```
postfix_expr
    = primary_expr
    = postfix_expr '++'
    = postfix_expr '--'
```

```
unary_expr
    = postfix_expr
    = '++' unary_expr
    = '--' unary_expr
    = unary_operator unary_expr
    = Identifier '(' ' ' ')'
    = Identifier '(' c_argument_expr_list ')'
    = 'CURRENT_ROLE'
```

```
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.6.33 C-Grammer

```

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
    = ExprInteger

```

```

c_real
    = ExprReal

c_string
    = c_string_literal

c_string_literal
    = string_literal

c_reference
    = reference

c_postfix_expr
    = c_primary_expr
    = c_postfix_expr '[' c_expr ']'
    = Identifier '(' ')'
    = Identifier '(' c_argument_expr_list ')'
    = c_postfix_expr '.' Identifier
    = c_postfix_expr '.' selector
    = c_postfix_expr '++'
    = c_postfix_expr '--'

c_argument_expr_list
    = c_assignment_expr
    = c_argument_expr_list ',' c_assignment_expr

c_unary_expr
    = c_postfix_expr
    = '++' c_unary_expr
    = '--' c_unary_expr
    = c_unary_operator c_postfix_expr

c_unary_operator
    = '+'
    = '-'
    = '~'
    = '!'

c_multiplicative_expr
    = c_unary_expr
    = c_multiplicative_expr '*' c_unary_expr
    = c_multiplicative_expr '/' c_unary_expr
    = c_multiplicative_expr '%' c_unary_expr

c_additive_expr
    = c_multiplicative_expr
    = c_additive_expr '+' c_multiplicative_expr
    = c_additive_expr '-' c_multiplicative_expr

c_shift_expr
    = c_additive_expr
    = c_shift_expr '<<' c_additive_expr
    = c_shift_expr '>>' c_additive_expr

c_relational_expr
    = c_shift_expr
    = c_relational_expr '<' c_shift_expr
    = c_relational_expr '>' c_shift_expr
    = c_relational_expr '<=' c_shift_expr
    = c_relational_expr '>=' c_shift_expr

c_equality_expr
    = c_relational_expr
    = c_equality_expr '==' c_relational_expr
    = c_equality_expr '!=' c_relational_expr

```

```

c_and_expr
    = c_equality_expr
    = c_and_expr '&' c_equality_expr

c_exclusive_or_expr
    = c_and_expr
    = c_exclusive_or_expr '^' c_and_expr

c_inclusive_or_expr
    = c_exclusive_or_expr
    = c_inclusive_or_expr '|' c_exclusive_or_expr

c_logical_and_expr
    = c_inclusive_or_expr
    = c_logical_and_expr '&&' c_inclusive_or_expr

c_logical_or_expr
    = c_logical_and_expr
    = c_logical_or_expr '||' c_logical_and_expr

c_conditional_expr
    = c_logical_or_expr
    = c_logical_or_expr '?' c_logical_or_expr ':' c_conditional_expr

c_assignment_expr
    = c_conditional_expr
    = c_unary_expr c_assignment_operator c_assignment_expr

c_assignment_operator
    = '='
    = '*='
    = '/='
    = '%='
    = '+='
    = '-='
    = '>>='
    = '<<='
    = '&='
    = '^='
    = '|='

c_expr
    = c_assignment_expr
    = c_expr ',' c_assignment_expr

c_constant_expr
    = c_conditional_expr

c_declaration
    = c_declaration_specifiers c_declarator_list ';'

c_declaration_specifiers
    = c_type_specifier
    = c_declaration_specifiers c_type_specifier

c_type_specifier
    = 'char'
    = 'short'
    = 'int'
    = 'long'
    = 'signed'
    = 'unsigned'
    = 'float'
    = 'double'
    = 'DD_STRING'

c_declarator_list

```



```

= c_declarator
= c_declarator_list ',' c_declarator

c_declarator
= Identifier
= Identifier c_declarator_array_specifier_list

c_declarator_array_specifier_list
= c_declarator_array_specifier
= c_declarator_array_specifier_list c_declarator_array_specifier

c_declarator_array_specifier
= '[' ']'
= '[' c_constant_expr ']'

c_statement
= c_labeled_statement
= c_compound_statement
= c_expr_statement
= c_selection_statement
= c_iteration_statement
= c_jump_statement

c_labeled_statement
= 'case' c_constant_expr ':' c_statement
= 'default' ':' c_statement

c_compound_statement
= '{' '}'
= '{' c_statement_list '}'
= '{' c_declaration_list '}'
= '{' c_declaration_list c_statement_list '}'

c_declaration_list
= c_declaration
= c_declaration_list c_declaration

c_statement_list
= c_statement
= c_statement_list c_statement

c_expr_statement
= ';'
= c_expr ';'

c_selection_statement
= 'if' '(' c_expr ')' c_statement
= 'if' '(' c_expr ')' c_statement 'else' c_statement
= 'switch' '(' c_expr ')' c_statement

c_iteration_statement
= 'do' c_statement 'while' '(' c_expr ')' ';'
= 'while' '(' c_expr ')' c_statement
= 'for' '(' ';' ';' ';' ')'
  c_statement
= 'for' '(' ';' ';' c_expr ')'
  c_statement
= 'for' '(' ';' c_expr ';' ')'
  c_statement
= 'for' '(' ';' c_expr ';' c_expr ')'
  c_statement
= 'for' '(' c_expr ';' ';' ')'
  c_statement
= 'for' '(' c_expr ';' ';' c_expr ')'
  c_statement
= 'for' '(' c_expr ';' c_expr ';' ')'
  c_statement

```

```

= 'for' '(' c_expr ';' c_expr ';' c_expr ')'
  c_statement

```

```

c_jump_statement
= 'continue' ';'
= 'break' ';'
= 'return' ';'
= 'return' c_expr ';'

```

A.6.34 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_type_redefinition
= array_size_redefinition
= help_redefinition
= required_label_redefinition
= response_codes_reference_redefinition

array_type_redefinition
= 'REDEFINE' array_type

array_size_redefinition
= 'REDEFINE' array_size

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
= optional_label_redefinition
= minimum_value_redefinition
= maximum_value_redefinition
= axis_scaling_redefinition
= help_redefinition
= constant_unit_redefinition

axis_scaling_redefinition
= 'DELETE' 'SCALING' ';'
= 'REDEFINE' axis_scaling

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_axis_items_redefinition

```

```

= block_a_characteristics_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
= block_a_help_redefinition
= block_a_image_items_redefinition
= block_a_item_array_items_redefinition
= block_a_label_redefinition
= block_a_local_parameters_redefinition
= block_a_list_items_redefinition
= block_a_menu_items_redefinition
= block_a_method_items_redefinition
= block_a_parameters_redefinition
= block_a_parameter_lists_redefinition
= block_a_refresh_items_redefinition
= block_a_source_items_redefinition
= block_a_unit_items_redefinition
= block_a_wao_items_redefinition
= block_a_waveform_items_redefinition
= block_b_number_redefinition
= block_b_type_redefinition

block_a_axis_items_redefinition
= 'DELETE' 'AXIS_ITEMS' ';'
= 'REDEFINE' block_a_axis_items

block_a_characteristics_redefinition
= 'REDEFINE' block_a_characteristics

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_help_redefinition
= help_redefinition

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_label_redefinition
    = required_label_redefinition

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_parameters_redefinition
    = 'PARAMETERS' '{' parameter_redefinition_list '}'

parameter_redefinition_list
    = parameter_redefinition
    = parameter_redefinition_list parameter_redefinition

parameter_redefinition
    = 'REDEFINE' member
    = 'ADD' member

block_a_parameter_lists_redefinition
    = 'PARAMETER_LISTS' '{' parameter_redefinition_list '}'

block_a_refresh_items_redefinition
    = 'DELETE' 'REFRESH_ITEMS' ';'
    = 'REDEFINE' block_a_refresh_items

block_a_source_items_redefinition
    = 'DELETE' 'SOURCE_ITEMS' ';'
    = 'REDEFINE' block_a_source_items

block_a_unit_items_redefinition
    = 'DELETE' 'UNIT_ITEMS' ';'
    = 'REDEFINE' block_a_unit_items

block_a_wao_items_redefinition
    = 'DELETE' 'WRITE_AS_ONE_ITEMS' ';'
    = 'REDEFINE' block_a_wao_items

block_a_waveform_items_redefinition
    = 'DELETE' 'WAVEFORM_ITEMS' ';'
    = 'REDEFINE' block_a_waveform_items

block_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
= optional_label_redefinition
= help_redefinition
= members_redefinition
= chart_height_redefinition
= chart_width_redefinition
= chart_type_redefinition
= chart_cycle_time_redefinition
= chart_length_redefinition

chart_height_redefinition
= 'DELETE' 'HEIGHT' ';'
= 'REDEFINE' chart_height

chart_width_redefinition
= 'DELETE' 'WIDTH' ';'
= 'REDEFINE' chart_width

chart_type_redefinition
= 'DELETE' 'TYPE' ';'
= 'REDEFINE' chart_type

chart_cycle_time_redefinition
= 'DELETE' 'CYCLE_TIME' ';'
= 'REDEFINE' chart_cycle_time

chart_length_redefinition
= 'DELETE' 'LENGTH' ';'
= 'REDEFINE' chart_length

collection_redefinition
= 'DELETE' 'COLLECTION' Identifier ';'
= 'REDEFINE' 'COLLECTION' Identifier '{' '}' = 'REDEFINE'
'COLLECTION' 'OF' item_type Identifier '{' '}'
= 'REDEFINE' 'COLLECTION' Identifier '{' collection_attribute_list '}'
= 'REDEFINE' 'COLLECTION' 'OF' item_type Identifier '{'
collection_attribute_list '}'
= 'COLLECTION' Identifier '{' collection_attribute_redefinition_list
'}'
= 'COLLECTION' 'OF' item_type Identifier '{'
collection_attribute_redefinition_list '}'

collection_attribute_redefinition_list
= collection_attribute_redefinition
= collection_attribute_redefinition_list

collection_attribute_redefinition
= help_redefinition
= optional_label_redefinition
= members_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_header_redefinition
= command_slot_redefinition
= command_index_redefinition
= command_block_redefinition

```

```

= command_number_redefinition
= command_operation_redefinition
= command_transaction_redefinition
= command_connection_redefinition
= command_module_redefinition
= command_response_codes_redefinition

command_header_redefinition
= 'DELETE' 'HEADER' ';'
= 'REDEFINE' command_header

command_slot_redefinition
= 'DELETE' 'SLOT' ';'
= 'REDEFINE' command_slot

command_index_redefinition
= 'DELETE' 'INDEX' ';'
= 'REDEFINE' command_index

command_block_redefinition
= 'DELETE' 'BLOCK' ';'
= 'REDEFINE' command_block

command_number_redefinition
= 'DELETE' 'NUMBER' ';'
= 'REDEFINE' command_number

command_operation_redefinition
= 'REDEFINE' command_operation

command_transaction_redefinition
= 'ADD' command_transaction
= 'REDEFINE' command_transaction
= 'DELETE' 'TRANSACTION' ';'
= 'DELETE' 'TRANSACTION' Integer ';'

command_connection_redefinition
= 'DELETE' 'CONNECTION' ';'
= 'REDEFINE' command_connection

command_module_redefinition
= 'DELETE' 'MODULE' ';'
= 'REDEFINE' command_module

command_response_codes_redefinition
= response_codes_reference_redefinition

connection_redefinition
= 'DELETE' 'CONNECTION' Identifier ';'
= 'REDEFINE' 'CONNECTION' Identifier '{' '}'
= 'REDEFINE' 'CONNECTION' Identifier '{'
  connection_attribute_list '}'
= 'CONNECTION' Identifier '{'
  connection_attribute_redefinition_list '}'

connection_attribute_redefinition_list
= connection_attribute_redefinition
= connection_attribute_redefinition_list

connection_attribute_redefinition
= appinstance_redefinition

appinstance_redefinition
= 'REDEFINE' appinstance

domain_redefinition

```

```

= 'DELETE DOMAIN' Identifier ';'
= 'REDEFINE DOMAIN' Identifier '{' domain_attribute_list '}'
= 'DOMAIN' Identifier '{' domain_attribute_redefinition_list '}'

domain_attribute_redefinition_list
= domain_attribute_redefinition
= domain_attribute_redefinition_list domain_attribute_redefinition

domain_attribute_redefinition
= handling_redefinition
= response_codes_reference_redefinition

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
= display_items_redefinition
= edit_items_redefinition
= required_label_redefinition
= post_edit_actions_redefinition
= pre_edit_actions_redefinition

display_items_redefinition
= 'DELETE' 'DISPLAY_ITEMS' ';'
= 'REDEFINE' display_items

edit_items_redefinition
= 'REDEFINE' edit_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
= optional_label_redefinition
= help_redefinition
= members_redefinition

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
= optional_label_redefinition
= help_redefinition
= members_redefinition
= graph_x_axis_redefinition

```

```

    = graph_height_redefinition
    = graph_width_redefinition

    = graph_cycle_time_redefinition

graph_x_axis_redefinition
    = 'DELETE' 'X_AXIS' ';'
    = 'REDEFINE' waveform_x_axis

graph_height_redefinition
    = 'DELETE' 'HEIGHT' ';'
    = 'REDEFINE' graph_height

graph_width_redefinition
    = 'DELETE' 'WIDTH' ';'
    = 'REDEFINE' graph_width

graph_cycle_time_redefinition
    = 'DELETE' 'CYCLE_TIME' ';'
    = 'REDEFINE' graph_cycle_time

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
    = optional_label_redefinition
    = help_redefinition
    = grid_vectors_redefinition
    = grid_handling_redefinition
    = grid_height_redefinition
    = grid_width_redefinition
    = grid_orientation_redefinition
    = grid_validity_redefinition

grid_vectors_redefinition
    = 'REDEFINE' grid_vectors

grid_handling_redefinition
    = 'DELETE' 'HANDLING' ';'
    = 'REDEFINE' grid_handling

grid_height_redefinition
    = 'DELETE' 'HEIGHT' ';'
    = 'REDEFINE' grid_height

grid_width_redefinition
    = 'DELETE' 'WIDTH' ';'
    = 'REDEFINE' grid_width

grid_orientation_redefinition
    = 'DELETE' 'ORIENTATION' ';'
    = 'REDEFINE' grid_orientation

grid_validity_redefinition
    = 'DELETE' 'VALIDITY' ';'
    = 'REDEFINE' grid_validity

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
    = optional_label_redefinition
    = help_redefinition
    = image_path_redefinition
    = image_link_redefinition
    = image_validity_redefinition

image_path_redefinition
    = 'REDEFINE' image_path

image_link_redefinition
    = 'DELETE' 'LINK' ';'
    = 'REDEFINE' image_link

image_validity_redefinition
    = 'DELETE' 'VALIDITY' ';'
    = 'REDEFINE' image_validity

item_array_redefinition
    = 'DELETE' 'ITEM_ARRAY' Identifier ';'
    = 'REDEFINE' 'ITEM_ARRAY' 'OF' item_type Identifier '{' '}'
    = 'REDEFINE' 'ITEM_ARRAY' 'OF' item_type Identifier '{'
item_array_attribute_list '}'
    = 'ITEM_ARRAY' 'OF' 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
    = elements_redefinition
    = help_redefinition
    = optional_label_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
    = help_redefinition
    = optional_label_redefinition
    = list_capacity_redefinition
    = list_count_redefinition

```

```
= list_type_redefinition

list_capacity_redefinition
= 'DELETE' 'CAPACITY' ';'
= 'REDEFINE' list_capacity

list_count_redefinition
= 'DELETE' 'COUNT' ';'
= 'REDEFINE' list_count

list_type_redefinition
= 'REDEFINE' list_type

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
= help_redefinition
= required_label_redefinition
= menu_access_redefinition
= menu_entry_redefinition
= menu_items_redefinition
= menu_purpose_redefinition
= menu_role_redefinition
= menu_style_redefinition
= validity_redefinition
= pre_edit_actions_redefinition
= pre_read_actions_redefinition
= pre_write_actions_redefinition
= post_edit_actions_redefinition
= post_read_actions_redefinition
= post_write_actions_redefinition

menu_access_redefinition
= 'DELETE' 'ACCESS' ';'
= 'REDEFINE' menu_access

menu_entry_redefinition
= 'DELETE' 'ENTRY' ';'
= 'REDEFINE' menu_entry

menu_items_redefinition
= 'REDEFINE' menu_items

menu_purpose_redefinition
= 'DELETE' 'PURPOSE' ';'
= 'REDEFINE' menu_purpose

menu_role_redefinition
= 'DELETE' 'ROLE' ';'
= 'REDEFINE' menu_role

menu_style_redefinition
= 'DELETE' 'STYLE' ';'
= 'REDEFINE' menu_style

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
= help_redefinition
= required_label_redefinition
= method_access_redefinition
= method_definition_redefinition
= validity_redefinition
= variable_class_redefinition

method_access_redefinition
= 'REDEFINE' method_access

method_definition_redefinition
= 'REDEFINE' method_definition

program_redefinition
= 'DELETE' 'PROGRAM' Identifier ';'
= 'REDEFINE' 'PROGRAM' Identifier '{' '}'
= 'REDEFINE' 'PROGRAM' Identifier '{' program_attribute_list '}'
= 'PROGRAM' Identifier '{' program_attribute_redefinition_list
  '}'

program_attribute_redefinition_list
= program_attribute_redefinition
= program_attribute_redefinition_list program_attribute_redefinition

program_attribute_redefinition
= arguments_redefinition
= response_codes_reference_redefinition

arguments_redefinition
= 'DELETE' 'ARGUMENTS' ';'
= 'REDEFINE' arguments

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
= help_redefinition
= required_label_redefinition
= members_redefinition
= response_codes_reference_redefinition

refresh_relation_redefinition
= 'DELETE' 'REFRESH' Identifier ';'
= 'REDEFINE' 'REFRESH' Identifier
  '{' '}'
= 'REDEFINE' 'REFRESH' Identifier
  '{' refresh_relation_left ':' refresh_relation_right '}'

unit_relation_redefinition
= 'DELETE' 'UNIT' Identifier ';'
= 'REDEFINE' 'UNIT' Identifier
  '{' '}'
= 'REDEFINE' 'UNIT' Identifier
  '{' unit_relation_left ':' unit_relation_right '}'

```

```

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

help_redefinition
= 'DELETE' 'HELP' ';'
= 'REDEFINE' help

required_label_redefinition
= 'REDEFINE' label

optional_label_redefinition
= 'DELETE' 'LABEL' ';'
= 'REDEFINE' label

response_codes_reference_redefinition
= 'DELETE' 'RESPONSE_CODES' ';'
= 'REDEFINE' response_codes

validity_redefinition
= 'DELETE' 'VALIDITY' ';'
= 'REDEFINE' validity

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

```

```

= help_redefinition
= optional_label_redefinition
=
= source_line_color_redefinition
= source_line_type_redefinition
= source_y_axis_redefinition
= members_redefinition
= source_refresh_actions_redefinition
= source_init_actions_redefinition
= source_exit_actions_redefinition

source_emphasis_redefinition
= 'DELETE' 'EMPHASIS' ';'
= 'REDEFINE' source_emphasis

source_line_color_redefinition
=
= 'DELETE' 'LINE_COLOR' ';'
= 'REDEFINE' source_line_color

source_line_type_redefinition
= 'DELETE' 'LINE_TYPE' ';'
= 'REDEFINE' source_line_type

source_y_axis_redefinition
= 'DELETE' 'Y_AXIS' ';'
= 'REDEFINE' waveform_y_axis

source_init_actions_redefinition
= 'DELETE' 'INIT_ACTIONS' ';'
= 'REDEFINE' source_init_actions

source_refresh_actions_redefinition
= 'DELETE' 'REFRESH_ACTIONS' ';'
= 'REDEFINE' source_refresh_actions

source_exit_actions_redefinition
= 'DELETE' 'REFRESH_ACTIONS' ';'
= 'REDEFINE' source_refresh_actions

variable_redefinition
= 'DELETE' 'VARIABLE' Identifier ';'
= 'REDEFINE' 'VARIABLE' Identifier '{' variable_attribute_list
  '}'
= 'VARIABLE' Identifier '{' variable_attribute_redefinition_list
  '}'

variable_attribute_redefinition_list
=
= variable_attribute_redefinition_list variable_attribute_redefinition

variable_attribute_redefinition
= constant_unit_redefinition
= default_value_redefinition
= handling_redefinition
= help_redefinition
= initial_value_redefinition
= required_label_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
= refresh_actions_redefinition
= read_timeout_redefinition
= response_codes_reference_redefinition
= type_redefinition

```

```
= validity_redefinition
= variable_class_redefinition
= variable_style_redefinition
= write_timeout_redefinition

variable_class_redefinition
= 'REDEFINE' variable_class

variable_style_redefinition
= 'DELETE' 'STYLE' ';'
= 'REDEFINE' variable_style

constant_unit_redefinition
= 'DELETE' 'CONSTANT_UNIT' ';'
= 'REDEFINE' constant_unit

default_value_redefinition
= 'DELETE' 'DEFAULT_VALUE' ';'
= 'REDEFINE' default_value

handling_redefinition
= 'DELETE' 'HANDLING' ';'
= 'REDEFINE' handling

initial_value_redefinition
= 'DELETE' 'INITIAL_VALUE' ';'
= 'REDEFINE' initial_value

pre_edit_actions_redefinition
= 'DELETE' 'PRE_EDIT_ACTIONS' ';'
= 'REDEFINE' pre_edit_actions

post_edit_actions_redefinition
= 'DELETE' 'POST_EDIT_ACTIONS' ';'
= 'REDEFINE' post_edit_actions

pre_read_actions_redefinition
= 'DELETE' 'PRE_READ_ACTIONS' ';'
= 'REDEFINE' pre_read_actions

post_read_actions_redefinition
= 'DELETE' 'POST_READ_ACTIONS' ';'
= 'REDEFINE' post_read_actions

pre_write_actions_redefinition
= 'DELETE' 'PRE_WRITE_ACTIONS' ';'
= 'REDEFINE' pre_write_actions

post_write_actions_redefinition
= 'DELETE' 'POST_WRITE_ACTIONS' ';'
= 'REDEFINE' post_write_actions

refresh_actions_redefinition
= 'DELETE' 'REFRESH_ACTIONS' ';'
= 'REDEFINE' refresh_actions

read_timeout_redefinition
= 'DELETE' 'READ_TIMEOUT' ';'
= 'REDEFINE' read_timeout

write_timeout_redefinition
= 'DELETE' 'WRITE_TIMEOUT' ';'
= 'REDEFINE' write_timeout

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

arithmetic_type_redefinition
= 'INTEGER' arithmetic_option_size_redefinition '{'
  arithmetic_option_redefinition_list '}'
= 'UNSIGNED_INTEGER' arithmetic_option_size_redefinition '{'
  arithmetic_option_redefinition_list '}'
= 'FLOAT' '{' arithmetic_option_redefinition_list '}'
= 'DOUBLE' '{' arithmetic_option_redefinition_list '}'

arithmetic_option_size_redefinition
= arithmetic_option_size_redefinition1

arithmetic_option_size_redefinition1
=
= '(' Integer ')'

arithmetic_option_redefinition_list
= arithmetic_option_redefinition
= arithmetic_option_redefinition_list arithmetic_option_redefinition

arithmetic_option_redefinition
= arithmetic_default_value_redefinition
= arithmetic_initial_value_redefinition
= display_format_redefinition
= edit_format_redefinition
= enumerator_list_redefinition
= maximum_value_redefinition
= maximum_value_n_redefinition
= minimum_value_redefinition
= minimum_value_n_redefinition
= scaling_factor_redefinition

arithmetic_default_value_redefinition
= 'DELETE' 'DEFAULT_VALUE' ';'
= 'REDEFINE' arithmetic_default_value

arithmetic_initial_value_redefinition
= 'DELETE' 'INITIAL_VALUE' ';'
= 'REDEFINE' arithmetic_initial_value

display_format_redefinition
= 'DELETE' 'DISPLAY_FORMAT' ';'
= 'REDEFINE' display_format

edit_format_redefinition
= 'DELETE' 'EDIT_FORMAT' ';'
= 'REDEFINE' edit_format

enumerator_list_redefinition
= 'DELETE' Integer ';'
= 'REDEFINE' enumerator
= 'ADD' enumerator

maximum_value_redefinition
= 'DELETE' 'MAX_VALUE' ';'
= 'REDEFINE' maximum_value

maximum_value_n_redefinition
= 'DELETE' MAX_VALUE_n ';'
= 'REDEFINE' maximum_value_n

minimum_value_redefinition
= 'DELETE' 'MIN_VALUE' ';'

```

```

    = 'REDEFINE' minimum_value

minimum_value_n_redefinition
    = 'DELETE' MIN_VALUE_n ';'
    = 'REDEFINE' minimum_value_n

scaling_factor_redefinition
    = 'DELETE' 'SCALING_FACTOR' ';'
    = 'REDEFINE' scaling_factor

enumerated_type_redefinition
    = 'ENUMERATED' enumerated_option_size_redefinition '{'
      enumerated_option_redefinition_list '}'
    = 'BIT_ENUMERATED' enumerated_option_size_redefinition '{'
      bit_enumerated_option_redefinition_list '}'

enumerated_option_size_redefinition
    = enumerated_option_size_redefinition1

enumerated_option_size_redefinition1
    =
    = '(' Integer ')'

enumerated_option_redefinition_list
    = enumerated_option_redefinition
    = enumerated_option_redefinition_list enumerated_option_redefinition

enumerated_option_redefinition
    = arithmetic_default_value_redefinition
    = arithmetic_initial_value_redefinition
    = enumerators_redefinition

enumerators_redefinition
    = 'DELETE' Integer ';'
    = 'REDEFINE' enumerator
    = 'ADD' enumerator

bit_enumerated_option_redefinition_list
    = bit_enumerated_option_redefinition
    = bit_enumerated_option_redefinition_list

bit_enumerated_option_redefinition
    = arithmetic_default_value_redefinition
    = arithmetic_initial_value_redefinition
    = bit_enumerators_redefinition

bit_enumerators_redefinition
    = 'DELETE' Integer ';'
    = 'REDEFINE' bit_enumerator
    = 'ADD' bit_enumerator

string_type_redefinition
    = string_option_type '{' string_option_redefinition_list '}'
    = string_option_type string_option_size_redefinition '{'
string_option_redefinition_list '}'

string_option_size_redefinition
    = string_option_size_redefinition1

string_option_size_redefinition1
    = '(' Integer ')'

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_list_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_list_redefinition
    = 'DELETE' string ';'
    = 'REDEFINE' enumerator
    = 'ADD' enumerator

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
    = help_redefinition
    = optional_label_redefinition
    = members_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
    = help_redefinition
    = optional_label_redefinition
    = waveform_type_redefinition
    = handling_redefinition
    = waveform_emphasis_redefinition
    = waveform_exit_actions_redefinition
    = waveform_line_color_redefinition
    = waveform_line_type_redefinition
    = waveform_y_axis_redefinition
    = waveform_init_actions_redefinition
    = waveform_refresh_actions_redefinition
    = waveform_key_points_redefinition

waveform_type_redefinition
    = 'REDEFINE' waveform_type
    = 'TYPE' waveform_type_redefinition

waveform_type_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_emphasis_redefinition
    = 'DELETE' 'EMPHASIS' ';'
    = 'REDEFINE' waveform_emphasis

waveform_exit_actions_redefinition
    = 'DELETE' 'EXIT_ACTIONS' ';'
    = 'REDEFINE' waveform_exit_actions

waveform_line_color_redefinition
    = 'DELETE' 'LINE_COLOR' ';'
    = 'REDEFINE' waveform_line_color

waveform_line_type_redefinition
    = 'DELETE' 'LINE_TYPE' ';'
    = 'REDEFINE' waveform_line_type

waveform_y_axis_redefinition
    = 'DELETE' 'Y_AXIS' ';'
    = 'REDEFINE' waveform_y_axis

waveform_init_actions_redefinition
    = 'DELETE' 'INIT_ACTIONS' ';'
    = 'REDEFINE' waveform_init_actions

waveform_refresh_actions_redefinition

```

```

= 'DELETE' 'REFRESH_ACTIONS' ';'
= 'REDEFINE' waveform_refresh_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_listR

key_points_options_redefinition_listR
= key_points_options_redefinition
= key_points_options_redefinition_listR key_points_options_redefinition

key_points_options_redefinition
= waveform_x_values_redefinition
= waveform_y_values_redefinition

```

A.6.35 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

selector
= 'CAPACITY'
= 'CONSTANT_UNIT'
= 'COUNT'
= 'DEFAULT_VALUE'
= 'FIRST'
= 'HELP'
= 'INITIAL_VALUE'
= 'LABEL'
= 'LAST'
= 'MAX_VALUE'
= 'MAX_VALUE n'
= 'MIN_VALUE'
= 'MIN_VALUE n'
= 'SCALING'
= 'SCALING_FACTOR'
= 'VARIABLE_STATUS'
= 'VIEW_MAX'
= 'VIEW_MIN'
= 'X_AXIS'
= 'Y_AXIS'

dictionary_reference
= '[' Identifier ']'

axis_reference
= reference_without_call

block_reference
= reference_without_call

chart_reference
= reference_without_call

collection_reference
= reference_without_call

connection_reference
= reference_without_call

edit_display_reference
= reference_without_call

```

```
file_reference
    = reference_without_call
graph_reference
    = reference_without_call
list_reference
    = reference_without_call
item_array_reference
    = reference_without_call
menu_reference
    = reference_without_call
method_reference
    = reference_without_call
record_reference
    = reference_without_call
refresh_reference
    = reference_without_call
response_code_reference
    = reference_without_call
source_reference
    = reference_without_call
unit_reference
    = reference_without_call
variable_reference
    = reference_without_call
wao_reference
    = reference_without_call
waveform_reference
    = reference_without_call
```

Annex B (normative)

EDDL Builtin library

B.1 General

This annex describes Builtin subroutines, which can be used in an EDD. The Builtins support user or device interactions. Typical user interactions are, for example, user inputs of VARIABLE values or acknowledgments. Typical device interactions are, for example, send commands to the device or interpreting response codes and status bytes. This annex contains the library of Builtins.

In addition to the Builtins specified in this document, user-specific Builtins can be used. These Builtins shall be integrated into the EDD application.

NOTE The Builtins are listed in alphabetical order.

B.2 Conventions for Builtin descriptions

The following conventions apply for the Builtin definitions.

- Table B.1 defines the format for the Builtins lexical element tables.
- Table B.2 defines the contents of the columns of Table B.1

Table B.1 – Format for the Builtins lexical element tables

Parameter Name	Type	Direction	Usage	Description

Table B.2 – Contents of the lexical element table

Column	Text	Meaning
Parameter name	<name>	is the name of the formal parameter
	"<return>"	represents the return value of the Builtin
Type	data type	abstract data type (basic or derived) of the formal parameter
Direction	I	specifies an input parameter
	O	specifies an output parameter
	I/O	specifies an input and output parameter
	—	to be used if there is no returned value (VOID)
Usage	m	formal parameter is mandatory
	o	formal parameter is optional
	c	formal parameter is conditional
	—	to be used if there is no returned value (VOID)
Description	<text>	descriptive text for the parameter
	—	to be used if there is no returned value (VOID)

B.3 Builtin abort

Purpose

Builtin abort will display a message indicating that the method has been aborted and wait for acknowledgement from the user. Once acknowledgement has been made, the system will execute any abort methods in the abort method list and will exit the method.

Lexical structure

abort (VOID)

The lexical elements of the Builtin abort are shown in Table B.3.

Table B.3 – Builtin abort

Parameter Name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.4 Builtin abort_on_all_comm_errors

Purpose

Builtin abort_on_all_comm_errors sets the method to abort upon receiving any communication error.

When a method is started, it does not abort on a communication error until abort_on_all_comm_errors is called. This frees a method from having to handle any communication error codes by aborting the method if it receives any communications error.

After this call, any Builtin call which makes a communication request that returns an error will cause the method to abort. A communication error of zero is not considered a communication error.

The communication error codes are defined by the consortia. Their related strings may be defined in a text dictionary.

Lexical structure

abort_on_all_comm_errors(VOID)

The lexical element of the Builtin abort_on_all_comm_errors is shown in Table B.4.

Table B.4 – Builtin abort_on_all_comm_errors

Parameter Name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.5 Builtin ABORT_ON_ALL_COMM_STATUS

Purpose

Builtin ABORT_ON_ALL_COMM_STATUS will set all of the bits in the comm status abort mask. This will cause the system to abort the current method if the device returns any comm status value. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is set.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks, and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_ALL_COMM_STATUS(VOID)

The lexical element of the Builtin ABORT_ON_ALL_COMM_STATUS is shown in Table B.5.

Table B.5 – Builtin ABORT_ON_ALL_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.6 Builtin ABORT_ON_ALL_DEVICE_STATUS

Purpose

Builtin ABORT_ON_ALL_DEVICE_STATUS will set all of the bits in the device status abort mask. This will cause the system to abort the current method if the device returns any device status value. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_ALL_DEVICE_STATUS (VOID)

The lexical element of the Builtin ABORT_ON_ALL_DEVICE_STATUS is shown in Table B.6.

Table B.6 – Builtin ABORT_ON_ALL_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.7 Builtin ABORT_ON_ALL_RESPONSE_CODES

Purpose

Builtin ABORT_ON_ALL_RESPONSE_CODES will set all of the bits in the response code abort mask. This will cause the system to abort the current method if the device returns any response code value.

The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_ALL_RESPONSE_CODES (VOID)

The lexical element of the Builtin ABORT_ON_ALL_RESPONSE_CODES is shown in Table B.7.

Table B.7 – Builtin ABORT_ON_ALL_RESPONSE_CODES

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.8 Builtin abort_on_all_response_codes

Purpose

The Builtin `abort_on_all_response_codes` causes the method to abort if it receives any non-zero response code, after calling a Builtin, which results in a request to a device. This Builtin frees a method from having to handle any error response codes. When a method is started, it does not abort on any response code until this Builtin is called. A response code is a value representing an application-specific error condition. A response code value of zero is not considered as an error response code. Each response code and response code type is defined in the EDDL.

Lexical structure

`abort_on_all_response_codes(VOID)`

The lexical element of the Builtin `abort_on_all_response_codes` is shown in Table B.8.

Table B.8 – Builtin abort_on_all_response_codes

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.9 Builtin abort_on_comm_error

Purpose

Causes the method to abort upon receiving a specific communications error.

The Builtin `abort_on_comm_error` causes the method to abort if it receives a specific communications error.

When a method is started, it does not abort on a specific communications error until Builtin `abort_on_comm_error` is called.

NOTE A communications error of zero is not considered a communications error.

This Builtin permits the method writer to select which communications errors should be handled by the method and it frees a method from having to handle a specific communications error code.

The communications error codes are defined by the consortia. Their related strings may be defined in a text dictionary.

Lexical structure

`abort_on_comm_error(error)`

The lexical elements of the Builtin `abort_on_comm_error` are shown in Table B.9.

Table B.9 – Builtin abort_on_comm_error

Parameter name	Type	Direction	Usage	Description
error	unsigned long	I	m	specifies the error code
<return>	long	O	m	is any of the return codes specified in Table B.235

B.10 Builtin ABORT_ON_COMM_ERROR

Purpose

Builtin ABORT_ON_COMM_ERROR will set the no comm error mask in such a way that the method will be aborted if a comm error is found while sending a command.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_COMM_ERROR(VOID)

The lexical element of the Builtin ABORT_ON_COMM_ERROR is shown in Table B.10.

Table B.10 – Builtin ABORT_ON_COMM_ERROR

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.11 Builtin ABORT_ON_COMM_STATUS

Purpose

The Builtin ABORT_ON_COMM_STATUS will set the correct bit(s) in the comm status abort mask in such a way that the specified comm_status_value will cause the method to abort. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 1.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send command function for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_COMM_STATUS(comm_status)

The lexical elements of the Builtin ABORT_ON_COMM_STATUS are shown in Table B.11.

Table B.11 – Builtin ABORT_ON_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
comm_status	unsigned char	l	m	specifies a bit mask for the communication status
<return>	VOID	—	—	—

B.12 Builtin ABORT_ON_DEVICE_STATUS

Purpose

The Builtin ABORT_ON_DEVICE_STATUS will set the correct bit(s) in the device status abort mask such that the specified device status value will cause the method to abort. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send command function for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_DEVICE_STATUS(device_status)

The lexical elements of the Builtin ABORT_ON_DEVICE_STATUS are shown in Table B.12.

Table B.12 – Builtin ABORT_ON_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
device_status	unsigned char	l	m	specifies a bit mask for the device status
<return>	VOID	—	—	—

B.13 Builtin ABORT_ON_NO_DEVICE

Purpose

The Builtin ABORT_ON_NO_DEVICE will set the no devices mask in such a way that the method will be aborted if no device is found while sending a command.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send command function for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

ABORT_ON_NO_DEVICE(VOID)

The lexical element of the Builtin ABORT_ON_NO_DEVICE is shown in Table B.13.

Table B.13 – Builtin ABORT_ON_NO_DEVICE

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.14 Builtin ABORT_ON_RESPONSE_CODE

Purpose

The Builtin ABORT_ON_RESPONSE_CODE will set the correct bit(s) in the response code abort mask such that the specified response code value will cause the method to abort. The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method.

The following is a list of all the available masks. The default values of the masks are as follows:

- response_code_abort_mask = ACCESS_RESTRICTED | CMD_NOT_IMPLEMENTED
- response_code_retry_mask = NO BITS SET
- cmd_48_response_code_abort_mask = ACCESS_RESTRICTED | CMD_NOT_IMPLEMENTED
- cmd_48_response_code_retry_mask = NO BITS SET
- device_status_abort_mask = DEVICE_MALFUNCTION
- device_status_retry_mask = NO BITS SET
- cmd_48_device_status_abort_mask = DEVICE_MALFUNCTION
- cmd_48_device_status_retry_mask = NO BITS SET
- comm_status_abort_mask = NO BITS SET
- comm_status_retry_mask = ALL BITS SET
- cmd_48_comm_status_abort_mask = NO BITS SET
- cmd_48_comm_status_retry_mask = ALL BITS SET
- no_device_abort_mask = ALL BITS SET
- comm_error_abort_mask = ALL BITS SET
- comm_error_retry_mask = NO BITS SET
- cmd_48_no_device_abort_mask = ALL BITS SET
- cmd_48_comm_error_abort_mask = ALL BITS SET
- cmd_48_comm_error_retry_mask = NO BITS SET
- cmd_48_data_abort_mask = NO BITS SET
- cmd_48_data_retry_mask = NO BITS SET

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

```
void ABORT_ON_RESPONSE_CODE(response code)
```

The lexical elements of the Builtin ABORT_ON_RESPONSE_CODE are shown in Table B.14.

Table B.14 – Builtin ABORT_ON_RESPONSE_CODE

Parameter name	Type	Direction	Usage	Description
response_code	int	l	m	specifies a bit mask for the response code
<return>	VOID	—	—	—

B.15 Builtin abort_on_response_code

Purpose

The Builtin abort_on_response_code causes the method to abort upon receiving the specified response code.

The Builtin causes the method to abort if it receives a specific response code. This Builtin frees a method from having to handle a specific response code.

When a method is started, it will not abort on a specific response code until this Builtin is called. A response code is a value representing an application-specific error condition.

NOTE A response code value of zero is not considered an error response code. Each response code type is defined in the EDD.

Lexical structure

abort_on_response_code (code)

The lexical elements of the Builtin abort_on_response_code are shown in Table B.15.

Table B.15 – Builtin abort_on_response_code

Parameter name	Type	Direction	Usage	Description
code	unsigned long	I	m	specifies a bit mask for the response code
<return>	long	O	m	is any of the return codes specified in Table B.235

B.16 Builtin abs

Purpose

The Builtin abs returns the absolute value of a floating point value.

Lexical structure

abs (x)

The lexical elements of the Builtin abs are shown in Table B.16.

Table B.16 – Builtin abs

Parameter name	Type	Direction	Usage	Description
x	double	I	m	a floating-point value
<return>	double	O	m	the absolute value

B.17 Builtin ACKNOWLEDGE

Purpose

The Builtin ACKNOWLEDGE will display the prompt and wait for the enter key to be pressed. Unlike Builtin acknowledge, the prompt may NOT contain embedded device VARIABLES.

Lexical structure

ACKNOWLEDGE (prompt)

The lexical elements of the Builtin ACKNOWLEDGE are shown in Table B.17.

Table B.17 – Builtin ACKNOWLEDGE

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be SUCCESS.

B.18 Builtin acknowledge

Purpose

The Builtin acknowledge will display the prompt and wait for acknowledgment from the user. The prompt may contain local and/or device variable values.

Lexical structure

acknowledge (prompt, global_var_ids)

The lexical elements of the Builtin acknowledge are shown in Table B.18.

Table B.18 – Builtin acknowledge

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
global_var_ids	array of int	I	m	specifies an array which contains unique identifiers of VARIABLE instance
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be SUCCESS.

B.19 Builtin acos

Purpose

The Builtin acos returns the arc cosine of a floating-point value.

Lexical structure

acos (x)

The lexical elements of the Builtin acos are shown in Table B.19.

Table B.19 – Builtin acos

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the arc cosine

B.20 Builtin add_abort_method (version A)

Purpose

The Builtin add_abort_method will add a method to the abort method list, which is the list of methods to be executed if the current method is aborted. The abort method list can hold up to 20 methods at any one time. The methods are run in the order they are added to the list, and the same method may be added to the list more than once. The list is cleared after each non_abort method is executed.

`add_abort_method` is only executed when the method is aborted, and not when the method exits under normal operating conditions. Methods can be aborted due to an abort mask condition when sending a command, or when the Builtin abort is called.

Lexical structure

```
add_abort_method(abort_method_id)
```

The lexical elements of the Builtin `add_abort_method` are shown in Table B.20.

Table B.20 – Builtin `add_abort_method`

Parameter name	Type	Direction	Usage	Description
<code>abort_method_id</code>	reference	I	m	specifies the name of a METHOD instance
<return>	int	O	m	specifies whether the method was successfully added to the list or if the list was full

NOTE The symbolic names for the return value should be TRUE or FALSE.

B.21 Builtin `add_abort_method` (version B)

Purpose

The Builtin `add_abort_method` adds an abort method to the calling method's abort method list. The abort method list is the list of methods, which will be executed if the method aborts. The abort method list applies only to the current execution of the method.

Methods abort when response codes or communication errors occur which have been set to trigger an abort via other Builtin calls, for example, `abort_on_all_comm_error` and `abort_on_all_response_codes`. Methods can also be explicitly aborted by a direct call to Builtin `method_abort`, or when an error occurs. The normal completion the execution of a method is not considered an abort.

The abort method list is initially empty when a method begins execution. The abort methods are added to the back of the list and are removed and run from the front of the list in the order they appear, i.e. first-in first-out. An abort method may appear more than once in the list.

Abort methods may be shared between methods, meaning two methods may have the same abort method in their respective abort method lists.

The `method_id` parameter is specified by the method writer using the Builtin `ITEM_ID` and the name of the method. Builtins abort methods cannot call `add_abort_method`.

Lexical structure

```
long
add_abort_method (method_id)
```

The lexical elements of the Builtin `add_abort_method` are shown in Table B.21.

Table B.21 – Builtin `add_abort_method`

Parameter name	Type	Direction	Usage	Description
<code>method_id</code>	unsigned long	I	m	specifies the ID of the method
<return>	long	O	m	is any of the return codes specified in Table B.235

B.22 Builtin asin

Purpose

The Builtin asin returns the arc sine of a floating point value.

Lexical structure

`asin(x)`

The lexical elements of the Builtin asin are shown in Table B.22.

Table B.22 – Builtin asin

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the arc sine

B.23 Builtin assign

Purpose

The Builtin assign sets the value of a VARIABLE to another VARIABLE's value. It sets the value of the destination VARIABLE equal to the value of the source VARIABLE.

NOTE The Builtin is a shorthand way of doing an ABN_GET_TYPE_VALUE followed by a put_type_value.

The parameters `dst_id`, `src_id`, `dst_member_id`, and `src_member_id` are specified using the Builtins ITEM_ID and MEMBER ID and the reference of the VARIABLE. Both the source VARIABLES and the destination VARIABLE shall have the same type.

Lexical structure

`long assign (dst_id, dst_member_id, src_id, src_member_id)`

The lexical elements of the Builtin assign are shown in Table B.23.

Table B.23 – Builtin assign

Parameter name	Type	Direction	Usage	Description
<code>dst_id</code>	unsigned long	I	m	specifies the item ID of the destination variable
<code>dst_member_id</code>	unsigned long	I	m	specifies the member ID of the destination variable
<code>src_id</code>	unsigned long	I	m	specifies the item ID of the source variable
<code>src_member_id</code>	unsigned long	I	m	specifies the member ID of the source variable
<return>	long	O	m	is any of the return codes specified in Table B.235

B.24 Builtin assign_double

Purpose

The Builtin assign_double assigns a value to a VARIABLE.

Lexical structure

`assign_double(device_var, new_value)`

The lexical elements of the Builtin assign_double are shown in Table B.24.

Table B.24 – Builtin assign_double

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	specifies the name of the variable
new_value	double	I	m	specifies the new value of the referenced variable
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE if the assignment was successful and FALSE if the variable identifier was invalid.

B.25 Builtin assign_float

Purpose

The Builtin assign_float assigns a value to VARIABLE.

Lexical structure

assign_float(device_var, new_value)

The lexical elements of the Builtin assign_float are shown in Table B.25.

Table B.25 – Builtin assign_float

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	specifies the name of the variable
new_value	double	I	m	specifies the new value of the referenced variable
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE if the assignment was successful and FALSE if the variable identifier was invalid.

B.26 Builtin assign_int

Purpose

The Builtin assign_int assigns a value to VARIABLE.

Lexical structure

assign_int(device_var, new_value)

The lexical elements of the Builtin assign_int are shown in Table B.26.

Table B.26 – Builtin assign_int

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	specifies the name of the variable
new_value	int	I	m	specifies the new value of the referenced variable
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE if the assignment was successful and FALSE if the variable identifier was invalid.

B.27 Builtin assign_var

Purpose

The Builtin assign_var assigns a value of the source VARIABLE to the destination VARIABLE. Both variables shall have the same type.

Lexical structure

assign_var(destination_var, source_var)

The lexical elements of the Builtin assign_var are shown in Table B.27.

Table B.27 – Builtin assign_var

Parameter name	Type	Direction	Usage	Description
destination_var	reference	I	m	specifies the name of the destination variable
source_var	reference	I	m	specifies the name of the source variable
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE if the assignment was successful and FALSE if the variable identifier was invalid

B.28 Builtin atan

Purpose

The Builtin atan returns the arc tangent of a floating point value.

Lexical structure

atan(x)

The lexical elements of the Builtin atan are shown in Table B.28.

Table B.28 – Builtin atan

Parameter Name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the arc tangent

B.29 Builtin ATOF

Purpose

The Builtin ATOF converts a string into a floating-point value.

Lexical structure

ATOF(string)

The lexical elements of the Builtin ATOF are shown in Table B.30.

Table B.29 – Builtin ATOF

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the string to convert
<return>	double	O	m	specifies the return value of the Builtin

B.30 Builtin atof

Purpose

This Builtin converts a text into a floating-point value.

Lexical structure

`atof(str)`

The lexical elements of the Builtin `atof` are shown in Table B.30.

Table B.30 – Builtin atof

Parameter name	Type	Direction	Usage	Description
str	char[]	I	m	specifies the text to convert
<return>	float	O	m	specifies the return value of the Builtin

B.31 Builtin ATOI

Purpose

The Builtin `atoi` converts a string into an integer value.

Lexical structure

`atoi(str)`

The lexical elements of the Builtin `atoi` are shown in Table B.32.

Table B.31 – Builtin ATOI

Parameter name	Type	Direction	Usage	Description
str	DD_STRING	I	m	specifies the string to convert
<return>	int	O	m	specifies the return value of the Builtin

B.32 Builtin atoi

Purpose

This Builtin converts a text into an integer value.

Lexical structure

`atoi(str)`

The lexical elements of the Builtin `atoi` are shown in Table B.32.

Table B.32 – Builtin atoi

Parameter name	Type	Direction	Usage	Description
str	char[]	I	m	specifies the text to convert
<return>	int	O	m	specifies the return value of the Builtin

B.33 Builtin BUILD_MESSAGE

Purpose

The Builtin BUILD_MESSAGE returns a string based on a format specification. Any variable reference contained in the format string will be expanded to the current value. Method local variables and DD variables can be referenced in the message parameter. For more information, refer to the Builtin PUT_MESSAGE.

Lexical structure

BUILD_MESSAGE(string)

The lexical elements of the Builtin BUILD_MESSAGE are shown in Table B.33.

Table B.33 – Builtin BUILD_MESSAGE

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the format specification
<return>	DD_STRING	O	m	returns the formatted string

B.34 Builtin cbrt

Purpose

The Builtin cbrt returns the cubic root of a floating-point value.

Lexical structure

cbrt(x)

Table B.34 – Builtin cbrt

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the cubic root

B.35 Builtin ceil

Purpose

The Builtin ceil returns the smallest integral value not less than a specified floating-point value.

Lexical structure

ceil(x)

The lexical elements of the Builtin ceil are shown in Table B.35.

Table B.35 – Builtin ceil

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the ceiling value

B.36 Builtin cos

Purpose

The Builtin cos returns the cosine of a floating point value.

Lexical structure

`cos (x)`

The lexical elements of the Builtin cos are shown in Table B.36.

Table B.36 – Builtin cos

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the cosine

B.37 Builtin cosh

Purpose

The Builtin cosh returns the hyperbolic cosine of a floating point value.

Lexical structure

`cosh (x)`

The lexical elements of the Builtin cosh are shown in Table B.37.

Table B.37 – Builtin cosh

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the hyperbolic cosine

B.38 Builtin dassign

Purpose

The Builtin dassign assigns a value to a VARIABLE.

Lexical structure

`dassign(device_var, new_value)`

The lexical elements of the Builtin dassign are shown in Table B.38.

Table B.38 – Builtin dassign

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	specifies the name of the variable
new_value	double	I	m	specifies the new value of the referenced variable
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE if the assignment was successful and FALSE if the variable identifier was invalid.

B.39 Builtin Date_to_DayOfMonth

Purpose

The Builtin Date_to_DayOfMonth calculates the day of the month from a given number of days since 1 January 1900.

Lexical structure

Date_to_DayOfMonth(date)

The lexical elements of the Builtin Date_to_DayOfMonth are shown in Table B.39.

Table B.39 – Builtin Date_to_DayOfMonth

Parameter name	Type	Direction	Usage	Description
date	long	I	m	specifies the number of days. The EDD application shall support an implicit typecast between long and DATE, so this Builtin will also accept DATE types
<return>	int	O	m	returns the day of month

B.40 Builtin Date_to_Month

Purpose

The Builtin Date_to_Month calculates the month from a given number of days since 1 January 1900.

Lexical structure

Date_to_Month(date)

The lexical elements of the Builtin Date_to_Month are shown in Table B.40.

Table B.40 – Builtin Date_to_Month

Parameter name	Type	Direction	Usage	Description
date	long	I	m	specifies the number of days. The EDD application shall support an implicit typecast between long and DATE, so this Builtin will also accept DATE types
<return>	int	O	m	returns the month

B.41 Builtin Date_to_Year

Purpose

The Builtin Date_to_Year calculates the year from a given number of days since 1 January 1900.

Lexical structure

Date_to_Year(date)

The lexical elements of the Builtin Date_to_Year are shown in Table B.41.

Table B.41 – Builtin Date_to_Year

Parameter name	Type	Direction	Usage	Description
date	long	I	m	specifies the number of days. The EDD application shall support an implicit typecast between long and DATE, so this Builtin will also accept DATE types
<return>	int	O	m	returns the month from 1 January 1900

B.42 Builtin DELAY

Purpose

The Builtin DELAY will display a prompt message and pause for the specified number of seconds. The prompt may contain local variable values (see Builtin put_message for lexical structure). The prompt may NOT contain device variable values. The delay time shall be a positive number.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

DELAY(delay_time, prompt)

The lexical elements of the Builtin DELAY are shown in Table B.42.

Table B.42 – Builtin DELAY

Parameter name	Type	Direction	Usage	Description
delay_time	unsigned integer	l	m	specifies the delay in number of seconds
prompt	char[]	l	m	specifies a message to be displayed
<return>	VOID	—	—	—

B.43 Builtin delay

Purpose

The Builtin delay will display a prompt message, and pause for the specified number of seconds. The prompt may contain local and/or device VARIABLE values (see put_message for Lexical Structure). The delay time shall be a positive number.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

delay(delay_time, prompt, global_var_ids)

The lexical elements of the Builtin delay are shown in Table B.43.

Table B.43 – Builtin delay

Parameter name	Type	Direction	Usage	Description
delay_time	unsigned integer	l	m	specifies the delay in number of seconds
prompt	char[]	l	m	specifies a message to be displayed
global_var_ids	array of int	l	m	specifies an array which contains unique identifiers of VARIABLE instance
<return>	VOID	—	—	—

B.44 Builtin DELAY_TIME

Purpose

The Builtin DELAY_TIME will pause for the specified number of seconds. The delay time shall be a positive number.

Lexical structure

DELAY_TIME(delay_time)

The lexical elements of the Builtin DELAY_TIME are shown in Table B.44.

Table B.44 – Builtin DELAY_TIME

Parameter name	Type	Direction	Usage	Description
delay_time	unsigned integer	I	m	specifies the delay in number of seconds
<return>	VOID	—	—	—

B.45 Builtin delayfor

Purpose

The Builtin delayfor prints a string on the application display and waits for the allotted amount of time to expire. VARIABLE values may be embedded in the message string with formatting information.

Lexical structure

delayfor(duration, prompt, ids, indices, id_count)

The lexical elements of the Builtin delayfor are shown in Table B.45.

Table B.45 – Builtin delayfor

Parameter name	Type	Direction	Usage	Description
duration	long	I	m	specifies the delay in number of seconds
prompt	char[]	I	m	specifies a message to be displayed
ids	array of unsigned long	I	m	specifies the item IDs of a VARIABLE instance used in the prompt message
indices	array of unsigned long	I	m	specifies the member IDs or array indices of the VARIABLES used in the prompt message. Set to zero if there is no member ID or array index
id_count	long	I	m	specifies the number of array elements in the arrays for ids and indices. Set to zero if the arrays for ids and indices are NULL
<return>	long	O	m	is any of the return codes specified in Table B.235

B.46 Builtin DICT_ID

Purpose

The Builtin DICT_ID can be used to specify a text dictionary ID for calls to get stddict_string. It converts a text dictionary name into a text dictionary ID. The Builtins processed by the EDD tool, generating the dictionary ID for the provided dictionary name. The value, which the EDD tool replaces for DICT_ID (name) is an unsigned long data type.

A text dictionary entry consists of three parts: a unique ID, a name, and a multilingual description. This function retrieves the unique ID using the text dictionary name.

Lexical structure

DICTIONARY (name)

The lexical elements of the Builtin DICTIONARY are shown in Table B.46.

Table B.46 – Builtin DICTIONARY

Parameter name	Type	Direction	Usage	Description
name	char[]	I	m	specifies the name of the dictionary
<return>	long	O	m	specifies dictionary ID for the named string

B.47 Builtin discard_on_exit

Purpose

The Builtin discard_on_exit sets the termination action to discard any changes made by the method to VARIABLE values stored in a caching EDD application VARIABLE table, which have not already been sent to the device. It causes the method to discard any changes to VARIABLE values in the VARIABLE table when the method ends.

When a method modifies a VARIABLE value in a caching EDD application, it is changing a resident copy of that value. This change is effective only for the life of the method. To change the value(s) in the device, the method shall send the value(s) to the device or call send_on_exit. To save the values only in the VARIABLE table (rather than send them to the device), call the Builtin save_on_exit.

After method termination, the changed values in the VARIABLE table return to the device values that existed before they were changed by the method.

All values that have been changed but not sent to the device are handled at the termination of the method by either discarding them, sending them to the device, or saving the changes in a way that survives the end of the method (but does not affect the values in the device).

This Builtin takes no action in a non-caching EDD application because all parameter values are written directly to the device.

If a method does not call discard_on_exit, save_on_exit, or send_on_exit before exiting, any values changed by the Builtin are discarded as if the Builtin discard_on_exit had been called.

Lexical structure

discard_on_exit(VOID)

The lexical element of the Builtin discard_on_exit is shown in Table B.47.

Table B.47 – Builtin discard_on_exit

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.48 Builtin display

Purpose

The Builtin display will display the specified message on the screen, continuously updating the dynamic VARIABLE values used in the string (see put_message for Lexical Structure). The VARIABLE value will continue to be updated until operator acknowledgement is made.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

`display(prompt, global_var_ids)`

The lexical elements of the Builtin display are shown in Table B.48.

Table B.48 – Builtin display

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
global_var_ids	array of int	I	m	specifies an array which contains unique identifiers of VARIABLE instance
<return>	VOID	—	—	—

B.49 Builtin display_builtin_error

Purpose

The Builtin display_builtin_error displays on the application display device the text associated with a specified Builtin error. The Builtin display does not return control to the method until the user acknowledges the message.

NOTE 1 This Builtin provides a means of debugging calls to Builtin routines, which are returning an error. It should never be used in a final production method, because the types of errors returned by Builtins should never be received in a fully debugged method.

NOTE 2 The Builtin error codes may be defined in a file.

Lexical structure

`display_builtin_error(error)`

The lexical elements of the Builtin display_builtin_error are shown in Table B.49.

Table B.49 – Builtin display_builtin_error

Parameter name	Type	Direction	Usage	Description
error	long	I	m	specifies a code for a Builtin error
<return>	long	O	m	is any of the return codes specified in Table B.235

B.50 Builtin display_comm_error

Purpose

The Builtin display_comm_error displays the text string associated with a communications error on the application display device.

NOTE 1 A communications error of zero is not considered a communications error.

NOTE 2 The Builtin does not return control to the method until the user acknowledges the message.

Internally, this Builtin calls the Builtin get_comm_error_string to get the associated text string and the Builtin get_acknowledgement to display the associated text string and wait for user acknowledgement.

The communications error codes are defined by the consortia. Their related strings may be defined in a text dictionary.

Lexical structure

```
display_comm_error(error)
```

The lexical elements of the Builtin display_comm_error are shown in Table B.50.

Table B.50 – Builtin display_comm_error

Parameter name	Type	Direction	Usage	Description
error	unsigned long	I	m	specifies a code for a communication error
<return>	long	O	m	is any of the return codes specified in Table B.235

B.51 Builtin display_comm_status**Purpose**

The Builtin display_comm_status will display the string associated with the specified value of the comm_status_value octet. The message will remain on the screen. The displayed string will remain on the screen until operator acknowledgement is made.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

```
display_comm_status(comm_status_value)
```

The lexical elements of the Builtin display_comm_status are shown in Table B.51.

Table B.51 – Builtin display_comm_status

Parameter name	Type	Direction	Usage	Description
comm_status_value	int	I	m	specifies a code for a communication error
<return>	VOID	—	—	—

B.52 Builtin display_device_status**Purpose**

The Builtin display_device_status will display the string associated with the specified value of the device_status octet. The displayed string will remain on the screen until operator acknowledgement is made.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

```
display_device_status(device_status_value)
```

The lexical elements of the Builtin display_device_status are shown in Table B.52.

Table B.52 – Builtin display_device_status

Parameter name	Type	Direction	Usage	Description
device_status_value	int	I	m	specifies a code for the device status
<return>	VOID	—	—	—

B.53 Builtin display_dynamics

Purpose

The Builtin display_dynamics continuously displays a message on the application output device. The Builtin updates the values specified in the message until interrupted by the user. Control is not returned to the method until the user interrupts. (Application code scans for the acknowledgment; the Builtin waits for the application code to receive the user intervention, check it, and return control to the Builtin.)

VARIABLE values may be embedded in the message string with formatting information. This call can be made for device VARIABLES or local method VARIABLES. The primary purpose of the call is to display device value information that the user wants to observe over a period of time.

Lexical structure

display_dynamics(prompt, ids, indices, id_count)

The lexical elements of the Builtin display_dynamics are shown in Table B.53.

Table B.53 – Builtin display_dynamics

Parameter name	Type	Direction	Usage	Description
prompt	char[]	l	m	specifies a message to be displayed
ids	array of unsigned long	l	m	specifies the item IDs of the VARIABLES used in the prompt message
indices	array of unsigned long	l	m	specifies the member IDs or array index of the VARIABLES used in the prompt message. Set to zero if there is no member ID or array index
id_count	long	l	m	specifies the number of array elements in the arrays for ids and indices. Set to zero if the arrays for ids and indices are NULL
<return>	long	o	m	is any of the return codes specified in Table B.235

B.54 Builtin display_message

Purpose

The Builtin display_message displays a message on the applications output device. It returns control to the method without requiring user acknowledgment.

Device, local method VARIABLE values, and labels for device VARIABLES may be embedded in the prompt string with formatting information. This call is used to inform the user of system status changes that require no action.

Lexical structure

display_message (prompt, ids, indices, id_count)

The lexical elements of the Builtin display_message are shown in Table 54.

Table B.54 – Builtin display_message

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
ids	array of unsigned long	I	m	specifies the item IDs of the VARIABLES used in the prompt message
indices	array of unsigned long	I	m	specifies the member IDs or array index of the VARIABLES used in the prompt message. Set to zero if there is no member ID or array index
id_count	long	I	m	specifies the number of array elements in the arrays for ids and indices. Set to zero if the arrays for ids and indices are NULL
<return>	long	O	m	is any of the return codes specified in Table B.235

B.55 Builtin display_response_code

Purpose

The Builtin display_response_code displays the description of a response code for an item. The display Builtin does not return control to the method until the user has acknowledged the message.

Internally, this Builtin calls the Builtin get_response_code_string to get the associated text string and the Builtin get_acknowledgement to display the associated text string and wait for user acknowledgement.

A response code is a value representing an application-specific error condition. A response code value of zero is not considered an error response code. Each response code and response code type is defined in the EDD.

The parameter id and member_id refer to a parameter, which may have a response code associated with it after it was last read or written. The parameter id and member_id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the VARIABLE in question.

Lexical structure

```
display_response_code(id, member_id, code)
```

The lexical elements of the Builtin display_response_code are shown in Table B.55.

Table B.55 – Builtin display_response_code

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the item ID of the item which generated the response code
member_id	unsigned long	I	m	specifies the member ID of the item which generated the response code
code	unsigned long	I	m	specifies the response code as received from Builtin get_response_code
<return>	long	O	m	is any of the return codes specified in Table B.235

B.56 Builtin display_response_status

Purpose

The Builtin display_response_status will display the string associated with the specified value of the response_code octet. The displayed string will remain on the screen until operator acknowledgement is made.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

display_response_status(response_code_value)

The lexical elements of the Builtin display_response_status are shown in Table B.56.

Table B.56 – Builtin display_response_status

Parameter Name	Type	Direction	Usage	Description
response_code_value	int	I	m	specifies the response code to display in a prompt
<return>	VOID	—	—	—

B.57 Builtin display_xmtr_status

Purpose

The Builtin display_xmtr_status will display the string associated with the specified value of the indicated transmitter VARIABLE. The displayed string will remain on the screen until operator acknowledgement is made.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

display_xmtr_status(xmtr_variable, response_code_value)

The lexical elements of the Builtin display_xmtr_status are shown in Table B.57.

Table B.57 – Builtin display_xmtr_status

Parameter Name	Type	Direction	Usage	Description
xmtr_variable	int	I	m	specifies a reference to a VARIABLE instance
response_code_value	int	I	m	specifies a response code value
<return>	VOID	—	—	—

B.58 Builtin edit_device_value

Purpose

The Builtin edit_device_value displays a prompt with a device value specified by the param_id and param_member_id. The user can edit the displayed value.

NOTE A device value may reside in the VARIABLE table of a caching EDD application, and the value may not be written to the device until a method terminates or the value is explicitly sent to the device by the method.

VARIABLE values may be embedded in the message string with formatting information. The application edits and verifies the values entered for the device value. This editing is based on VARIABLE arithmetic types defined in the original EDD for each VARIABLE. The EDDL attributes associated with VARIABLES may include

- HANDLING
- DISPLAY_FORMAT
- EDIT_FORMAT
- MIN_VALUE
- MAX_VALUE

Lexical structure

`edit_device_value(prompt, ids, indices, id_count, param_id, param_member_id)`

The lexical elements of the Builtin `edit_device_value` are shown in Table B.58.

Table B.58 – Builtin `edit_device_value`

Parameter name	Type	Direction	Usage	Description
<code>prompt</code>	<code>char[]</code>	<code>l</code>	<code>m</code>	specifies a message to be displayed
<code>ids</code>	array of unsigned long	<code>l</code>	<code>m</code>	specifies the item IDs of the VARIABLES used in the prompt message
<code>indices</code>	array of unsigned long	<code>l</code>	<code>m</code>	specifies the member IDs or array index of the VARIABLES used in the prompt message. Set to zero if there is no member ID or array index
<code>id_count</code>	<code>long</code>	<code>l</code>	<code>m</code>	specifies the number of array elements in the arrays for <code>ids</code> and <code>indices</code> . Set to zero if the arrays for <code>ids</code> and <code>indices</code> are NULL
<code>param_id</code>	unsigned long	<code>l</code>	<code>m</code>	specifies the item ID of the VARIABLE to be modified
<code>param_member_id</code>	unsigned long	<code>l</code>	<code>m</code>	specifies the member ID of the VARIABLE to be modified
<code><return></code>	<code>long</code>	<code>O</code>	<code>m</code>	is any of the return codes specified in Table B.235

B.59 Builtin `edit_local_value`

Purpose

The Builtin `edit_local_value` displays a prompt with the local method VARIABLE name specified by the parameter `local_var`. The `local_var` parameter is a null-terminated C string of characters. The user can edit the displayed value of the local method VARIABLE.

Variable values may be embedded in the message string with formatting information. The EDD application is responsible for preventing invalid values from being returned by performing checks based on VARIABLE type only. The method shall perform range checking on returned values.

Lexical structure

`edit_local_value(prompt, ids, indices, id_count, local_var)`

The lexical elements of the Builtin `edit_local_value` are shown in Table B.59.

Table B.59 – Builtin edit_local_value

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
ids	array of unsigned long	I	m	specifies the item IDs of the VARIABLES used in the prompt message
indices	array of unsigned long	I	m	specifies the member IDs or array index of the VARIABLES used in the prompt message. Set to zero if there is no member ID or array index
id_count	long	I	m	specifies the number of array elements in the arrays for ids and indices. Set to zero if the arrays for ids and indices are NULL
local_var	char[]	I	m	specifies the name of the local VARIABLE that may be modified
<return>	long	O	m	is any of the return codes specified in Table B.235

B.60 Builtin exp

Purpose

The Builtin exp returns the exponential of a floating-point value.

Lexical structure

exp (x)

The lexical elements of the Builtin exp are shown in Table B.60.

Table B.60 – Builtin exp

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the exponential value

B.61 Builtin ext_send_command

Purpose

The Builtin ext_send_command and has the same functionality as the send_command function, except that the command status and data octets returned from the device with command 48 are also returned.

The calling function is responsible for allocating the arrays for the data to be returned. The cmd_status, and more_data_status VARIABLES are pointers to 3-octet arrays, and the more_data_info VARIABLE is a pointer to an array whose size is equal to the number of data octets that can be returned by the device in command 48 (maximum 25). If the status bit indicating that more data is available is returned (status class MORE), command 48 is automatically issued. If command 48 was not issued, the more_data_status, and more_data_info octets are set to zero.

Lexical structure

ext_send_command(cmd_number, cmd_status, more_data_status, more_data_info)

The lexical elements of the Builtin ext_send_command are shown in Table B.61.

Table B.61 – Builtin ext_send_command

Parameter name	Type	Direction	Usage	Description
cmd_number	int	I	m	specifies the COMMAND number
cmd_status	char[]	I	m	specifies the COMMAND status
more_data_status	char[]	I	m	specifies if more status data is available
more_data_info	char[]	I	m	specifies if more info is available
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be BI_SUCESS if the command was successful, BI_COMM_ERROR if an error occurred sending the command, or BI_NO_DEVICE if no device was found.

B.62 Builtin ext_send_command_trans

Purpose

The Builtin ext_send_command_trans sends the command with the specified transaction to the device. This function is to be used to send commands that have been defined with multiple transactions.

Builtin ext_send_command_trans has the same functionality as the send_command function, except that the command status and data octets returned from the device with command 48 are also returned.

The calling function is responsible for allocating the arrays for the data to be returned. The cmd_status, and more data status variables are pointers to 3-octet arrays, and the more data info variable is a pointer to an array whose size is equal to the number of data octets that can be returned by the device in command 48 (maximum 25). If command 48 was not issued, the more_data_status, and more_data_info octets are set to zero.

Lexical structure

```
ext_send_command_trans(cmd_number, transaction, cmd_status, more_data_status,
more_data_info)
```

The lexical elements of the Builtin ext_send_command_trans are shown in Table B.62.

Table B.62 – Builtin ext_send_command_trans

Parameter name	Type	Direction	Usage	Description
cmd_number	int	I	m	specifies the COMMAND number
transaction	int	I	m	specifies the number of the TRANSACTION
cmd_status	char[]	I	m	specifies the COMMAND status
more_data_status	char[]	I	m	specifies if more status data is available
more_data_info	char[]	I	m	specifies if more info is available
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be BI_SUCESS if the command was successful, BI_COMM_ERROR if an error occurred sending the command, or BI_NO_DEVICE if no device was found.

B.63 Builtin fail_on_all_comm_errors

Purpose

The Builtin fail_on_all_comm_errors presents the Builtins to send a return code to the method instead of aborting the method or retrying the communications request when any communications error is received.

Instead of aborting the method or attempting to retry a failing communications request when any communications error occurs, the Builtin fail_on_all_comm_errors initializes the Builtins to send a communications error code to the method.

When a communications error occurs, the Builtin performing a communications request at the time returns the error BLTIN_FAIL_COMM_ERROR. The method can call get_comm_error and subsequently get_comm_error_string to obtain additional information about the error and later display the error string, if appropriate.

Lexical structure

fail_on_all_comm_errors (VOID)

The lexical element of the Builtin fail_on_all_comm_errors is shown in Table B.63.

Table B.63 – Builtin fail_on_all_comm_errors

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.64 Builtin fail_on_all_response_codes

Purpose

The Builtin fail_on_all_response_codes presents the Builtins to return a Builtin error code to the method instead of aborting the method or retrying when any response code is received.

Instead of aborting the method or attempting to retry the failed application request, the Builtin fail_on_all_response_codes initializes the Builtins to send a return code to the method when any response code is received.

When a response code error occurs, any Builtin performing a request at the time returns the error BLTIN_FAIL_RESPONSE_CODE. The method can call get_comm_error and subsequently get_comm_error_string to obtain additional information about the error, if appropriate.

A response code is a value representing an application-specific error condition.

NOTE A response code value of zero is not considered an error response code. Each response code and response code type is defined in the EDD.

Lexical structure

fail_on_all_response_codes (VOID)

The lexical element of the Builtin fail_on_all_response_codes is shown in Table B.64.

Table B.64 – Builtin fail_on_all_response_codes

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.65 Builtin fail_on_comm_error

Purpose

The Builtin fail_on_comm_error presents the Builtins to return a return code to the method instead of aborting the method or retrying when a communication error is received.

Instead of aborting the method or attempting to retry a failed communications request, the Builtin `fail_on_comm_error` initializes the Builtins to return a Builtin error code to the method when a communications error occurs.

When the specified communications error occurs, any Builtin performing a communications request at the time returns the error Builtin `fail_on_comm_error`. The method can call `get_comm_error` and then `get_comm_error_string` to obtain additional information about the error and later display it, if appropriate.

The communications error codes are defined by the consortia. Their related strings may be defined in a text dictionary. A communications error of zero is not considered a communications error.

Lexical structure

```
fail_on_comm_error(error)
```

The lexical elements of the Builtin `fail_on_comm_error` are shown in Table B.65.

Table B.65 – Builtin `fail_on_comm_error`

Parameter name	Type	Direction	Usage	Description
error	unsigned long	I	m	specifies a communication error code
<return>	long	O	m	is any of the return codes specified in Table B.235

B.66 Builtin `fail_on_response_code`

Purpose

The Builtin `fail_on_response_code` sets the Builtins to return a Builtin error code to the method instead of aborting the method or retrying when a specified error response code is received.

Instead of aborting the method or retrying a failing request, the Builtin `fail_on_response_code` initializes the Builtins to return a return code to the method when a response code error occurs.

When the response code error occurs, any Builtin performing a request at the time returns the error `BLTIN_FAIL_RESPONSE_CODE`. The method can call `get_comm_error` and subsequently `get_comm_error_string` to obtain additional information about the error and later display it, if appropriate.

A response code is a value representing an application-specific error condition.

NOTE A response code value of zero is not considered an error response code. Each response code and response code type is defined in the EDD.

Lexical structure

```
fail_on_response_code(code)
```

The lexical elements of the Builtin `fail_on_response_code` are shown in Table B.66.

Table B.66 – Builtin `fail_on_response_code`

Parameter name	Type	Direction	Usage	Description
code	unsigned long	I	m	specifies the response code
<return>	long	O	m	is any of the return codes specified in Table B.235

B.67 Builtin fassign

Purpose

The Builtin fassign assigns a value to a VARIABLE.

Lexical structure

```
fassign(device_var, new_value)
```

The lexical elements of the Builtin fassign are shown in Table B.67.

Table B.67 – Builtin fassign

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	specifies the name of the VARIABLE
new_value	double	I	m	specifies the new value of the referenced VARIABLE
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE if the assignment was successful, and FALSE if the variable identifier was invalid.

B.68 Builtin fgetval

Purpose

The Builtin fgetval is specifically designed to be used in pre-read/write and post-read/write methods. This function will return the value of a device VARIABLE as it was received from the connected device. Scaling operations may then be performed on the VARIABLE value before it is stored in the EDD application.

Lexical structure

```
fgetval(VOID)
```

The lexical element of the Builtin fgetval is shown in Table B.68.

Table B.68 – Builtin fgetval

Parameter name	Type	Direction	Usage	Description
<return>	double	O	m	specifies the value of the variable

B.69 Builtin float_value

Purpose

The Builtin float_value returns the value of the specified VARIABLE.

Lexical structure

```
float_value(source_var_name)
```

The lexical elements of the Builtin float_value are shown in Table B.69.

Table B.69 – Builtin float_value

Parameter name	Type	Direction	Usage	Description
source_var_name	reference	I	m	specifies the name of a VARIABLE reference
<return>	double	O	m	specifies the value of the variable

B.70 Builtin floor

Purpose

The Builtin floor returns the largest integral value not greater than a specified floating point value.

Lexical structure

`floor(x)`

The lexical elements of the Builtin floor are shown in Table B.70.

Table B.70 – Builtin floor

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the floor value

B.71 Builtin fmod

Purpose

The Builtin fmod returns the floating point remainder of the division of two floating point values.

Lexical structure

`fmod(x, y)`

The lexical elements of the Builtin fmod are shown in Table B.71.

Table B.71 – Builtin fmod

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies the numerator
y	double	I	m	specifies the denominator
<return>	double	O	m	returns the remainder

B.72 Builtin fsetval

Purpose

The Builtin fsetval sets the value of a VARIABLE of type float for which a pre/post action has been designated to the specified value.

The function fsetval is specifically designed to be used in pre-read/write and post-read/write methods. This function will set the value of a device VARIABLE to the specified value.

Lexical structure

`fsetval(value)`

The lexical elements of the Builtin fsetval are shown in Table B.72.

Table B.72 – Builtin fsetval

Parameter name	Type	Direction	Usage	Description
value	double	I	m	specifies the new value for the VARIABLE
<return>	double	O	m	specifies the value of the variable

B.73 Builtin ftoa

Purpose

This Builtin converts a floating point value into a text string.

Lexical structure

`ftoa(f)`

The lexical elements of the Builtin ftoa are shown in Table B.73.

Table B.73 – Builtin ftoa

Parameter name	Type	Direction	Usage	Description
f	float	I	m	specifies a floating-point value to convert to a text
<return>	char[]	O	m	specifies the returned text

B.74 Builtin fvar_value

Purpose

The Builtin fvar_value returns the value of the specified VARIABLE.

Lexical structure

`fvar_value(source_var)`

The lexical elements of the Builtin fvar_value are shown in Table B.74.

Table B.74 – Builtin fvar_value

Parameter name	Type	Direction	Usage	Description
source_var	reference	I	m	specifies the name of the VARIABLE reference
<return>	double	O	m	specifies the value of the variable

B.75 Builtin get_acknowledgement

Purpose

The Builtin get_acknowledgement displays the provided prompt message on the application display device and returns to the method after the user acknowledges the message. Variable values may be embedded in the message string with formatting information.

Lexical structure

```
get_acknowledgement(prompt, ids, indices, id_count)
```

The lexical elements of the Builtin `get_acknowledgement` are shown in Table B.75.

Table B.75 – Builtin `get_acknowledgement`

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
ids	array of unsigned long	I	m	specifies the item IDs of the VARIABLES used in the prompt message
indices	array of unsigned long	I	m	specifies the member IDs or array indices of the VARIABLES used in the prompt message. Set to zero if there is no member ID or array indices
id_count	long	I	m	specifies the number of array elements in the arrays for ids and indices. Set to zero if the arrays for ids and indices are NULL
<return>	long	O	m	is any of the return codes specified in Table B.235

B.76 Builtin `get_comm_error`

Purpose

The Builtin `get_comm_error` returns the communications error code received from an application in the event of a Builtin return code of `BLTIN_FAIL_COMM_ERROR`.

This Builtin is used primarily for debugging purposes to determine what type of communications failures are occurring by displaying the communications error for the method writer to see.

In rare cases, the method could react to specific communications errors. However, the method should specify alternate handling of communications errors by way of the following:

- `fail_on_all_comm_errors`
- `fail_on_comm_error`
- `abort_on_all_comm_errors`
- `ABORT_ON_COMM_ERROR`

The alternate handling may cause the method to abort or immediately process the error based on the presence of a communications error.

The communications error codes are defined by the consortia. Their related strings may be defined in a text dictionary. A communications error of zero is not considered a communications error. The Builtin gets the communications error code returned from the last communications request.

Lexical structure

```
get_comm_error (VOID)
```

The lexical element of the Builtin `get_comm_error` is shown in Table B.76.

Table B.76 – Builtin get_comm_error

Parameter name	Type	Direction	Usage	Description
<return>	unsigned long	O	m	specifies a communication error code

B.77 Builtin get_comm_error_string

Purpose

The Builtin get_comm_error_string formats a descriptive string describing the communications error based on the error code that is provided. If possible, the description of the communications error is used; if the description is not available, a default communications error message with the error number is used.

Lexical structure

```
get_comm_error_string(error, string, maxlen)
```

The lexical elements of the Builtin get_comm_error_string are shown in Table B.77.

Table B.77 – Builtin get_comm_error_string

Parameter name	Type	Direction	Usage	Description
error	unsigned long	I	m	specifies the communication error code
string	char[]	I	m	specifies a description of the communication error
maxlen	long	I	m	specifies the maximal length of string
<return>	long	O	m	is any of the return codes specified in Table B.235

B.78 Builtin get_date

Purpose

The Builtin get_date retrieves the current unscaled date value of the VARIABLE being operated on. The Builtin is called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action.

The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a VARIABLE, which has one of these actions defined as an attribute, is read or written by an application, a user method, or an edit method.

Lexical structure

```
get_date(data, size)
```

The lexical elements of the Builtin get_date are shown in Table B.78.

Table B.78 – Builtin get_date

Parameter name	Type	Direction	Usage	Description
data	char[]	I	m	specifies the returned value of the VARIABLE. The structure of this octet buffer is defined in this specification. This output parameters shall point to valid storage
size	long	I	m	specifies the returned length of the VARIABLE. This output parameter shall point to valid storage
<return>	long	O	m	is any of the return codes specified in Table B.235

B.79 Builtin get_date_value**Purpose**

The Builtin get_date_value retrieves the current scaled value of the specified VARIABLE from the VARIABLE table (caching EDD application) or device and returns it.

The value is an octet buffer that may contain up to eight characters and a VARIABLE of device type date-and-time, time, or duration.

Lexical structure

get_date_value(id, member_id, data, size)

The lexical elements of the Builtin get_date_value are shown in Table B.79.

Table B.79 – Builtin get_date_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies an ID of a VARIABLE reference
member_id	unsigned long	I	m	specifies a member ID of a BLOCK_A, PARAMETER_LIST, RECORD, VALUE_ARRAY, COLLECTION or REFERENCE_ARRAY
data	char[]	I	m	specifies a buffer of max. 8 octet for the result of the Builtin
size	long	I	m	specifies the size of the used buffer
<return>	long	O	m	is any of the return codes specified in Table B.235

B.80 Builtin get_dds_error**Purpose**

The Builtin get_dds_error provides a means of debugging calls to Builtin routines, which are returning an error. It should never be used in a final production method, because the types of errors that are returned with get_dds_error should never be received in a fully debugged method.

A method may determine the actual EDD error received after the error return BLTIN_DDS_ERROR is received by calling this Builtin. Optionally, an allocated character pointer with its length may be passed to get the text from the error.

The EDD error and optional description returned are for the most recent EDD error.

Returns the EDD error generated by an earlier call, and optionally returns a text string describing the error.

Lexical structure

`get_dds_error(string, maxlen)`

The lexical elements of the Builtin `get_dds_error` are shown in Table B.80.

Table B.80 – Builtin `get_dds_error`

Parameter name	Type	Direction	Usage	Description
string	char[]	I	m	specifies a pointer to a text description of the error
maxlen	long	I	m	specifies the maximum length of the optional output buffer (string) in octets. This parameter is meaningful only if a parameter string has been allocated for the error message
<return>	long	O	m	specifies an error code generated by an earlier Builtin call

B.81 Builtin `GET_DEV_VAR_VALUE`

Purpose

The Builtin `GET_DEV_VAR_VALUE` will display the specified prompt message, and allow the user to edit the value of a device VARIABLE. The edited copy of the device VARIABLE value will be updated when the new value is entered, but will not be sent to the device. This shall be done explicitly using one of the send command functions.

The prompt may NOT contain embedded local and/or device VARIABLE values.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

`GET_DEV_VAR_VALUE(prompt, device_var_name)`

The lexical elements of the Builtin `GET_DEV_VAR_VALUE` are shown in Table B.81.

Table B.81 – Builtin `GET_DEV_VAR_VALUE`

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
device_var_name	reference	I	m	specifies the referenced name of the VARIABLE, which can be edited
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be `BI_SUCCESS`, if the variable was successfully modified, or `BI_ERROR`, if an error occurred entering the new value or accessing the specified variable.

B.82 Builtin get_dev_var_value

Purpose

The Builtin `get_dev_var_value` will display the specified prompt message and allow the user to edit the value of a device VARIABLE. The edited copy of the device VARIABLE value will be updated when the new value is entered but will not be sent to the device. This shall be done explicitly using one of the send command functions.

The prompt may contain embedded local and/or device VARIABLE values (see Builtin `PUT_MESSAGE`).

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

`get_dev_var_value(prompt, global_var_ids, device_var_name)`

The lexical elements of the Builtin `get_dev_var_value` are shown in Table B.82.

Table B.82 – Builtin `get_dev_var_value`

Parameter name	Type	Direction	Usage	Description
prompt	char[]	l	m	specifies a message to be displayed
global_var_ids	array of int	l	m	specifies an array which contains unique identifiers of VARIABLE instance
device_var_name	reference	l	m	specifies a name of the VARIABLE, which can be edited
<return>	int	o	m	specifies a status if the VARIABLE was successfully modified or if an error occurred entering the new value or accessing the specified VARIABLE

NOTE The symbolic name for the return value should be `BI_SUCCESS`, if the variable was successfully modified, or `BI_ERROR`, if an error occurred entering the new value or accessing the specified variable.

B.83 Builtin get_dictionary_string

Purpose

The Builtin `get_dictionary_string` will retrieve the text dictionary string associated with the given name in the current language. If the string is not available in the current language, the English string will be retrieved. If the string is not defined in either language, an error condition occurs, and the function will return `FALSE`. If the string is longer than the `max_str_len`, the string will be truncated. The parameter `dict_string_name` is the name of the string as it appears in the text dictionary.

Lexical structure

`get_dictionary_string(dict_string_name, string, max_str_len)`

The lexical elements of the Builtin `get_dictionary_string` are shown in Table B.83.

Table B.83 – Builtin get_dictionary_string

Parameter name	Type	Direction	Usage	Description
dict_string_name	reference	I	m	specifies the name of a dictionary entry
string	char[]	I	m	specifies a buffer for the result string
max_str_len	int	I	m	specifies the maximum length of the buffer
<return>	int	O	m	specifies a status if successful

NOTE The symbolic name for the return value should be TRUE, if successful; FALSE, if string could not be found.

B.84 Builtin get_double

Purpose

The Builtin get_double retrieves the current unscaled double value of the VARIABLE being operated on.

Gets the current value of the double VARIABLE as if it was directly read from a device.

The Builtin get_double is called only by methods associated with a pre-edit, post-edit, post-read or pre-write action.

The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a VARIABLE, which has one of these actions defined as an attribute, is read by an application or another method.

Lexical structure

get_double(value)

The lexical elements of the Builtin get_double are shown in Table B.84.

Table B.84 – Builtin get_double

Parameter name	Type	Direction	Usage	Description
value	double	O	m	specifies the returned value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.85 Builtin get_double_value

Purpose

The Builtin get_double_value retrieves the current scaled value of the specified VARIABLE from the VARIABLE table (caching EDD application) or device and returns it.

The parameters id and member_id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the variable.

The VARIABLE shall have a data type of double and be a valid variable in the VARIABLE table (caching EDD application) or device.

Lexical structure

```
get_double_value(id, member_id, value)
```

The lexical elements of the Builtin `get_double_value` are shown in Table B.85.

Table B.85 – Builtin `get_double_value`

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the Item ID of the VARIABLE to access
member_id	unsigned long	I	m	specifies the member ID of the VARIABLE to access
value	double	O	m	specifies the returned value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.86 Builtin `get_enum_string`

Purpose

The Builtin `get_enum_string` returns the string associated with the value of an enumerated or bit enumerated variable.

Lexical structure

```
get_enum_string(variable, value, string)
```

The lexical elements of the Builtin `get_enum_string` are shown in Table B.86.

Table B.86 – Builtin `get_enum_string`

Parameter name	Type	Direction	Usage	Description
variable	DD_ITEM	I	m	specifies an enumerated or bit enumerated variable
value	int	I	m	specifies a value of the variable
string	DD_STRING	O	m	returns string associated with the value of the variable
<return>	int	O	m	is any of the return codes specified in Table B.235

B.87 Builtin `get_float`

Purpose

The Builtin `get_float` retrieves the current unscaled floating-point value of the VARIABLE being operated on.

Gets the current value of an un-scaled floating-point VARIABLE as if it was directly read from a device.

The Builtin `get_float` can be called only by methods associated with a pre-edit, post-edit post-read, or pre-write action. The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a VARIABLE, which has one of these actions defined as an attribute, is read by an application, a user method, or an edit method.

Lexical structure

`get_float(value)`

The lexical elements of the Builtin `get_float` are shown in Table B.87.

Table B.87 – Builtin `get_float`

Parameter name	Type	Direction	Usage	Description
value	float	O	m	specifies the returned value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.88 Builtin `get_float_value`

Purpose

The Builtin `get_float_value` retrieves the current scaled value of the specified VARIABLE from the VARIABLE table (caching EDD application) or device and returns it.

The parameters `id` and `member id` are specified using the Builtins `ITEM_ID` and `MEMBER_ID` and the name or reference of the VARIABLE.

Lexical structure

`get_float_value(id, member_id, value)`

The lexical elements of the Builtin `get_float_value` are shown in Table B.88.

Table B.88 – Builtin `get_float_value`

Parameter name	Type	Direction	Usage	Description
id	unsigned long	O	m	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	m	specifies the member ID of the VARIABLE to access
value	float	O	m	specifies the returned value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.89 Builtin `GET_LOCAL_VAR_VALUE`

Purpose

The Builtin `GET_LOCAL_VAR_VALUE` will display the specified prompt message and allow the user to edit the value of a local VARIABLE.

The prompt may NOT contain embedded local and/or device VARIABLE values.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

`GET_LOCAL_VAR_VALUE(prompt, local_var_name)`

The lexical elements of the Builtin `GET_LOCAL_VAR_VALUE` are shown in Table B.89.

Table B.89 – Builtin GET_LOCAL_VAR_VALUE

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
local_var_name	reference d	I	m	specifies the name of the local VARIABLE to edit
<return>	int	O	m	specifies a status if the VARIABLE was successfully modified or if an error occurred entering the new value or accessing the specified VARIABLE

NOTE The symbolic name for the return value should be BI_SUCCESS, if the variable was successfully modified, or BI_ERROR, if an error occurred entering the new value or accessing the specified variable.

B.90 Builtin get_local_var_value

Purpose

The Builtin get_local_var_value will display the specified prompt message and allow the user to edit the value of a local VARIABLE.

The prompt may contain embedded local and/or device VARIABLE values.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

```
get_local_var_value(prompt, global_var_ids, local_var_name)
```

The lexical elements of the Builtin get_local_var_value are shown in Table B.90.

Table B.90 – Builtin get_local_var_value

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
global_var_ids	array of int	I	o	specifies an array which contains unique identifiers of VARIABLE instance
local_var_name	reference d	I	o	specifies the name of the local VARIABLE to edit
<return>	int	O	m	specifies a status if the VARIABLE was successfully modified or if an error occurred entering the new value or accessing the specified VARIABLE

NOTE The symbolic name for the return value should be BI_SUCCESS, if the variable was successfully modified, or BI_ERROR, if an error occurred entering the new value or accessing the specified variable.

B.91 Builtin get_more_status

Purpose

The Builtin get_more_status will issue a command 48 to the device and return the response code, communications status, and command status octets in the status array provided. It will also return any data octets returned in the more_data_info array. This function will process the status/data octets in the same manner as the send command.

The calling function is responsible for allocating the arrays for the data to be returned. The parameter `more_data_status` is a 3-octet array, and the `more_data_info` is an array whose size is equal to the number of data octets that can be returned by the device in command 48 (maximum 25).

Lexical structure

`get_more_status(more_data_status, more_data_info)`

The lexical elements of the Builtin `get_more_status` are shown in Table B.91.

Table B.91 – Builtin `get_more_status`

Parameter name	Type	Direction	Usage	Description
<code>more_data_status</code>	<code>char[]</code>	I	m	specifies if more info is available
<code>more_data_info</code>	<code>char[]</code>	I	m	specifies if more info is available
<return>	<code>int</code>	O	m	specifies a status if the command was successfully, if an error occurred or no device was found

NOTE The symbolic name for the return value should be `BI_SUCCESS`, if the command was successful, `BI_COMM_ERROR`, if an error occurred sending the command, or `BI_NO_DEVICE` if no device was found.

B.92 Builtin `get_resolve_status`

Purpose

The Builtin `get_resolve_status` returns the error number of the most recent “`get_resolve_`” built-ins. The reference resolution Builtins, those starting with “`get_resolve_`”, resolve the possibly dynamic ID of an element of a `BLOCK_A`, `PARAMETER`, `PARAMETER_LIST`, `VALUE_ARRAY`, `REFERENCE_ARRAY`, `COLLECTION`, or `RECORD`. If any of these resolve reference Builtins encounters an error, it returns a zero for the ID.

If the calling method wants to know what the error was, then it calls the Builtin `get_resolve_status` to get the error number. This Builtins used for debugging purposes when creating methods.

Returns the codes for errors generated by Builtin `resolve_array_ref`, `resolve_block_ref`, `resolve_parm_ref`, `resolve_param_list_ref` or `resolve_record_ref`.

Lexical structure

`get_resolve_status(VOID)`

The lexical element of the Builtin `get_resolve_status` is shown in Table B.92.

Table B.92 – Builtin `get_resolve_status`

Parameter name	Type	Direction	Usage	Description
<return>	unsigned long	O	m	specifies an error number

B.93 Builtin get_response_code

Purpose

The Builtin `get_response_code` returns the response code received from a device in the event of a Builtin communications call returning the error Builtin `fail_on_response_code`. It also returns the value reference (`err_id`, `err_member_id`) information associated with the original request. The original request is the last Builtin call which caused communication.

A response code is a value representing an application-specific error condition.

NOTE A response code value of zero is not considered an error response code. Each response code and response code type is defined in the EDD.

Gets the response code returned on the last communications request, and the item ID and member ID for the value being accessed when the response code was returned.

Lexical structure

```
get_response_code(resp_code, err_id, err_member_id)
```

The lexical elements of the Builtin `get_response_code` are shown in Table B.93.

Table B.93 – Builtin get_response_code

Parameter name	Type	Direction	Usage	Description
<code>resp_code</code>	unsigned long	I	m	specifies the number of the returned response code
<code>err_id</code>	unsigned long	O	m	specifies the returned item ID of the item generating the response code
<code>err_member_id</code>	unsigned long	O	m	specifies the returned member ID of the item causing a response code
<return>	long	O	m	is any of the return codes specified in Table B.235

B.94 Builtin get_response_code_string

Purpose

The Builtin `get_response_code_string` searches the response codes of the specified VARIABLE and returns the description of the specified response code. If the response code is not found, and if the item is a member of a RECORD or an element of an REFERENCE_ARRAY, then the RECORD or REFERENCE_ARRAY is checked for the specified code.

The method developer calls this Builtin if the description of a response code is unknown and shall be displayed. Otherwise, if the response code is known, the method can determine what to do in the presence of certain errors. Also non-standard error descriptions can be displayed by an application instead of the standard response codes.

Before calling this Builtin, the method shall use the Builtin `get_response_code` to get the `item_ID` and `member_ID` for the item with a response code. The parameters `id` and `member_id` are specified using the Builtins `ITEM_ID` and `MEMBER_ID` and the name or reference of the VARIABLE.

Lexical structure

```
get_response_code_string(id, member_id, code, string, maxlen)
```

The lexical elements of the Builtin `get_response_code_string` are shown in Table B.94.

Table B.94 – Builtin get_response_code_string

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the item ID of the item to find
member_id	unsigned long	I	m	specifies the member ID of the item to find
code	unsigned long	I	m	specifies the response code for which the descriptive string is to be found
string	char*	O	m	specifies the buffer for the returned response code description
maxlen	long	I	m	specifies the maximum length of the buffer
<return>	long	O	m	is any of the return codes specified in Table B.235

B.95 Builtin get_signed**Purpose**

The Builtin get_signed retrieves the current unscaled signed value of the VARIABLE being operated on.

Gets the current value of a signed VARIABLE as if it was directly read from a device.

The Builtin get_signed can be called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action. The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a VARIABLE, which has one of these actions defined as an attribute, is read by an application, a user method, or an edit method.

Lexical structure

```
get_signed(value)
```

The lexical elements of the Builtin get_signed are shown in Table B.95.

Table B.95 – Builtin get_signed

Parameter name	Type	Direction	Usage	Description
value	long	I	m	specifies the returned value of the signed VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.96 Builtin get_signed_value**Purpose**

The Builtin get_signed_value retrieves the current scaled value of the specified VARIABLE from the VARIABLE table (caching EDD application) or device and returns it.

The parameters id and member id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the VARIABLE. The VARIABLE shall have a data type of signed integer and shall be a valid VARIABLE in the VARIABLE table or device.

Lexical structure

```
get_signed_value(id, member_id, value)
```

The lexical elements of the Builtin `get_signed_value` are shown in Table B.96.

Table B.96 – Builtin `get_signed_value`

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	m	specifies the member ID of the VARIABLE to access.
value	long	O	m	specifies the returned value of the signed VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.97 Builtin `get_status_code_string`

Purpose

The Builtin `get_status_code_string` will return status code string for the specified device VARIABLE and status code value. If the string is longer than the maximum length defined in `status_string_length`, the string is truncated. The status code value shall be valid for the specified device VARIABLE.

Lexical structure

```
get_status_code_string(variable_name, status_code, status_string, status_string_length)
```

The lexical elements of the Builtin `get_status_code_string` are shown in Table B.97.

Table B.97 – Builtin `get_status_code_string`

Parameter name	Type	Direction	Usage	Description
variable_name	char[]	I	m	specifies the name of the VARIABLE to access
status_code	int	I	m	specifies the value of interesting status code
status_string	char *	O	m	specifies the buffer of the returned value of the signed VARIABLE
status_string_length	int	I	m	specifies the maximum size of the buffer
<return>	VOID	—	—	—

B.98 Builtin `get_status_string`

Purpose

The Builtin `get_status_string` searches the members of the referenced ENUMERATED or BIT_ENUMERATED VARIABLE and returns the description of the member matching the status in the given buffer.

The Builtin gets the string associated with an ~, ENUMERATED or BITE NUMERATED value. The most common use of ENUMERATED and BIT_ENUMERATED VARIABLES is to get a string associated with a status bit defined in the device for a particular status. The strings associated with ENUMERATED and BIT_ENUMERATED VARIABLES are defined in the EDDL.

The parameters `id` and `member_id` are specified using the Builtins `ITEM_ID` and `MEMBER_ID` and the name or reference of the VARIABLE in question.

Gets the description of a specific status in a specific device.

Lexical structure

`get_status_string (id, member_id, status, string, maxlen)`

The lexical elements of the Builtin `get_status_string` are shown in Table B.98.

Table B.98 – Builtin `get_status_string`

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	m	specifies the member ID of the VARIABLE to access
status	unsigned long	I	m	specifies the value to search
string	char*	O	m	specifies the buffer for the returned string
maxlen	long	I	m	specifies the maximum size of the buffer
<return>	long	O	m	is any of the return codes specified in Table B.235

B.99 Builtin `get_stddict_string`

Purpose

The Builtin `get_stddict_string` takes a text dictionary identifier and returns the corresponding text string in the supplied string buffer. The returned string is in the correct language for the application.

A text dictionary entry is composed of three parts: a unique id, a name, and a multilingual description. This Builtin returns the multilingual description portion of the entry. The Builtin strings are defined in the standard.dct file.

Lexical structure

`get_stddict_string(id, string, maxlen)`

The lexical elements of the Builtin `get_stddict_string` are shown in Table B.99.

Table B.99 – Builtin `get_stddict_string`

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the text dictionary identifier
string	char*	O	m	specifies the returned buffer for the string
maxlen	long	I	m	specifies the maximum size of the buffer
<return>	long	O	m	is any of the return codes specified in Table B.235

B.100 Builtin `get_string`

Purpose

The Builtin `get_string` retrieves the current, unscaled, string value of the VARIABLE being operated on.

Gets the current value of the string VARIABLE as if it was directly read from a device. The VARIABLE shall be a valid parameter. The VARIABLE shall have a data type of char and be an octet buffer which can contain various non-ANSI data types such as:

- ASCII
- PASSWORD
- EUC (Extended Unit Code)
- BITSTRING

Device VARIABLES of these data types shall be handled by the method because they are not supported by ANSI-C. The Builtin get_string is called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action. The method associated with pre-edit, post-edit, post-read, or post-write actions are run when a VARIABLE, which has one of these actions defined as an attribute, is read by an application, a user method, or an edit method.

Lexical structure

```
get_string (string, len)
```

The lexical elements of the Builtin get_string are shown in Table B.100.

Table B.100 – Builtin get_string

Parameter name	Type	Direction	Usage	Description
string	char*	O	m	specifies the buffer for the returned string
len	long	I	m	specifies the maximum size of the string. The returned length of the string
<return>	int	O	m	is any of the return codes specified in Table B.235

B.101 Builtin get_string_value

Purpose

The Builtin get_string_value retrieves the current scaled value of the specified VARIABLE from the VARIABLE table (caching EDD application) or device and returns it.

The parameters id and member id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the VARIABLE in question.

Lexical structure

```
get_string_value(id, member_id, string, len)
```

The lexical elements of the Builtin get_string_value are shown in Table B.101.

Table B.101 – Builtin get_string_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the ID of a VARIABLE reference
member_id	unsigned long	I	m	specifies the member ID of a BLOCK_A, PARAMETER_LIST, RECORD, VALUE_ARRAY, COLLECTION or REFERENCE_ARRAY
string	char*	O	m	specifies the buffer of result string
len	long	O	m	specifies the size of the result string
<return>	long	O	m	is any of the return codes specified in Table B.235

B.102 Builtin GET_TICK_COUNT

Purpose

The Builtin GET_TICK_COUNT returns the time in ms since the last system boot. It can be used for time stamps.

Lexical structure

GET_TICK_COUNT (VOID)

The lexical element of the Builtin GET_TICK_COUNT is shown in Table B.102.

Table B.102 – Builtin GET_TICK_COUNT

Parameter name	Type	Direction	Usage	Description
<return>	long	O	m	specifies the time in ms since the last system boot

B.103 Builtin get_unsigned

Purpose

The Builtin get_unsigned retrieves the current unscaled unsigned value of the VARIABLE being operated on.

The Builtin get_unsigned can be called only by methods associated with a pre-edit, post-edit, post-read, or pre-write method. The method associated with pre-edit, post-edit, post-read, or pre-write actions are run when a VARIABLE, which has one of these actions defined as an attribute, is read by an application, a user method, or an edit method.

Gets the current value of an unsigned variable as if it was directly read from a device.

Lexical structure

get_unsigned(value)

The lexical elements of the Builtin get_unsigned are shown in Table B.103.

Table B.103 – Builtin get_unsigned

Parameter name	Type	Direction	Usage	Description
value	unsigned long	O	m	specifies the returned value
<return>	long	O	m	is any of the return codes specified in Table B.235

B.104 Builtin get_unsigned_value

Purpose

The Builtin get_unsigned_value retrieves the current scaled value of the specified VARIABLE from the VARIABLE table (caching EDD application) or device and returns it.

The parameters id and member-id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the variable in question. The variable shall have a data type of unsigned. An unsigned value may contain device variables of the following data types:

- UNSIGNED
- ENUMERATED
- BIT_ENUMERATED
- INDEX

Device variables with data types of enumerated, bit_enumerated, and index shall be handled by the method because they are not supported by ANSI-C. The variable shall be a valid parameter.

Lexical structure

```
get_unsigned_value(id, member_id, value)
```

The lexical elements of the Builtin get_unsigned_value are shown in Table B.104.

Table B.104 – Builtin get_unsigned_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the item ID of the variable to access
member_id	unsigned long	I	m	specifies the member ID of the variable to access
value	unsigned long	O	m	specifies the returned value of the variable
<return>	long	O	m	is any of the return codes specified in Table B.235

B.105 Builtin iassign

Purpose

The Builtin iassign will assign the specified value to the device variable. The variable shall be valid, and shall reference a variable of type int.

Lexical structure

```
iassign (device_var, new_value)
```

The lexical elements of the Builtin iassign are shown in Table B.105.

Table B.105 – Builtin iassign

Parameter name	Type	Direction	Usage	Description
device_var	reference	I	m	specifies the name of the variable to access
new_value	int	I	m	specifies the new value of the VARIABLE
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE, if the assignment was successful, and FALSE, if the variable identifier was invalid.

B.106 Builtin igetval

Purpose

The Builtin igetval is specifically designed to be used in pre-read/write and post-read/write methods. This function will return the value of a device variable as it was received from the connected device. Scaling operations may then be performed on the variable value before it is stored in the EDD application.

Lexical structure

igetval (VOID)

The lexical element of the Builtin igetval is shown in Table B.106.

Table B.106 – Builtin igetval

Parameter name	Type	Direction	Usage	Description
<return>	int	O	m	specifies the value of the variable

B.107 Builtin IGNORE_ALL_COMM_STATUS

Purpose

The Builtin IGNORE_ALL_COMM_STATUS will clear all of the bits in the comm status retry and abort masks. This will cause the system to ignore all bits in the comm status value. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is set.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the Builtin send_command for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

IGNORE_ALL_COMM_STATUS (VOID)

The lexical element of the Builtin IGNORE_ALL_COMM_STATUS is shown in Table B.107.

Table B.107 – Builtin IGNORE_ALL_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.108 Builtin IGNORE_ALL_DEVICE_STATUS

Purpose

The Builtin IGNORE_ALL_DEVICE_STATUS will clear all of the bits in the device status retry and abort masks. This will cause the system to ignore all bits in the device status octet. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the Builtin send_command for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

IGNORE_ALL_DEVICE_STATUS (VOID)

The lexical element of the Builtin IGNORE_ALL_DEVICE_STATUS is shown in Table B.108.

Table B.108 – Builtin IGNORE_ALL_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.109 Builtin IGNORE_ALL_RESPONSE_CODES

Purpose

The Builtin IGNORE_ALL_RESPONSE_CODES will clear all of the bits in the response code retry and abort masks. This will cause the system to ignore all response code values returned from the device.

The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the Builtin send_command for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

IGNORE_ALL_RESPONSE_CODES (VOID)

The lexical element of the Builtin IGNORE_ALL_RESPONSE_CODES is shown in Table B.109.

Table B.109 – Builtin IGNORE_ALL_RESPONSE_CODES

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.110 Builtin IGNORE_COMM_ERROR**Purpose**

The Builtin IGNORE_COMM_ERROR will set the comm error mask such that a communications error condition will be ignored while sending a command.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

IGNORE_COMM_ERROR(VOID)

The lexical element of the Builtin IGNORE_COMM_ERROR is shown in Table B.110.

Table B.110 – Builtin IGNORE_COMM_ERROR

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.111 Builtin IGNORE_COMM_STATUS**Purpose**

The Builtin IGNORE_COMM_STATUS will clear the correct bit(s) in the comm status abort and retry mask such that the specified bits in the comm_status value will be ignored. comm_status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 1.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

IGNORE_COMM_STATUS(comm_status)

The lexical elements of the Builtin IGNORE_COMM_STATUS are shown in Table B.111.

Table B.111 – Builtin IGNORE_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
comm_status	int	l	m	specifies the new retry and abort mask for ignoring
<return>	VOID	—	—	—

B.112 Builtin IGNORE_DEVICE_STATUS**Purpose**

The Builtin IGNORE_DEVICE_STATUS will clear the correct bit(s) in the device status abort and retry masks such that the specified bits in the device status octet are ignored. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

IGNORE_DEVICE_STATUS(device_status)

The lexical elements of the Builtin IGNORE_DEVICE_STATUS are shown in Table B.112.

Table B.112 – Builtin IGNORE_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
device_status	int	l	m	specifies the new retry and abort mask for ignoring
<return>	VOID	—	—	—

B.113 Builtin IGNORE_NO_DEVICE**Purpose**

The Builtin IGNORE_NO_DEVICE will set the no device mask to show that the no-device condition should be ignored while sending a command.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

IGNORE_NO_DEVICE(VOID)

The lexical element of the Builtin IGNORE_NO_DEVICE is shown in Table B.113.

Table B.113 – Builtin IGNORE_NO_DEVICE

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.114 Builtin IGNORE_RESPONSE_CODE

Purpose

The Builtin IGNORE_RESPONSE_CODE will clear the correct bit(s) in the response code masks such that the specified response code value will be ignored. The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

```
void IGNORE_RESPONSE_CODE(response_code)
```

The lexical elements of the Builtin IGNORE_RESPONSE_CODE are shown in Table B.114.

Table B.114 – Builtin IGNORE_RESPONSE_CODE

Parameter name	Type	Direction	Usage	Description
response_code	int	—	m	specifies the new retry and abort mask for ignoring
<return>	VOID	—	—	—

B.115 Builtin int_value

Purpose

The Builtin int_value will return the value of the specified device variable. The variable identifier shall be valid and of type integer.

Lexical structure

```
int_value(source_var_name)
```

The lexical elements of the Builtin int_value are shown in Table B.115.

Table B.115 – Builtin int_value

Parameter name	Type	Direction	Usage	Description
source_var_name	char[]	I	m	specifies the name of the VARIABLE
<return>	int	O	m	specifies the value of the variable

B.116 Builtin is_NaN

Purpose

The Builtin is_NaN checks whether a double floating-point value is not a number. Not a number is specified by the IEEE 754 specification as an invalid double value. The value for NaN for a double value is: 0x7FF8000000000000

Lexical structure

is_NaN(dvalue)

The lexical elements of the Builtin is_NaN are shown in Table B.116.

Table B.116 – Builtin is_NaN

Parameter name	Type	Direction	Usage	Description
dvalue	double	I	m	specifies the name of the variable
<return>	long	O	m	is one if the dvalue is equal to NaN, zero otherwise

B.117 Builtin isetval

Purpose

The Builtin isetval will set the value of the device variable of type float for which a pre/post action has been designated to the specified value. isetval is specifically designed to be used in pre-read/write and post-read/write methods. This function will set the value of a device variable to the specified value.

Lexical structure

isetval(value)

The lexical elements of the Builtin isetval are shown in Table B.117.

Table B.117 – Builtin isetval

Parameter name	Type	Direction	Usage	Description
value	int	I	m	specifies the new value of the variable
<return>	int	O	m	specifies the value of the variable

B.118 Builtin ITEM_ID

Purpose

The Builtin ITEM_ID can be used in methods to specify the item ID of a variable for calls to other.

Lexical structure

ITEM_ID(name)

The lexical elements of the Builtin ITEM_ID are shown in Table B.118.

Table B.118 – Builtin ITEM_ID

Parameter name	Type	Direction	Usage	Description
name	reference	I	m	specifies the name of the variable
<return>	unsigned long	O	m	specifies the VARIABLES item ID

B.119 Builtin ITOA

Purpose

The Builtin ITOA converts a floating-point value into a string.

Lexical structure

ITOA(i)

The lexical elements of the Builtin ITOA are shown in Table B.119.

Table B.119 – Builtin ITOA

Parameter name	Type	Direction	Usage	Description
i	int	I	m	specifies an integer value to convert to a string
<return>	DD_STRING	O	m	specifies the value converted to a string

B.120 Builtin itoa

Purpose

This Builtin converts a floating-point value into a text string.

Lexical structure

itoa(i)

The lexical elements of the Builtin itoa are shown in Table B.120.

Table B.120 – Builtin itoa

Parameter name	Type	Direction	Usage	Description
i	int	I	m	specifies an integer value to convert to a text
<return>	char[]	O	m	specifies the returned text

B.121 Builtin ivar_value

Purpose

The Builtin ivar_value will return the value of the specified variable. The variable source_var_name shall be valid and of type integer.

Lexical structure

ivar_value(source_var_name)

The lexical elements of the Builtin ivar_value are shown in Table B.121.

Table B.121 – Builtin ivar_value

Parameter name	Type	Direction	Usage	Description
source_var_name	reference	I	m	specifies the name of the variable
<return>	int	O	m	specifies the value of the variable

B.122 Builtin lassign

Purpose

The Builtin lassign will assign the specified value to the device variable. The variable shall be valid, and shall reference a variable of type long.

Lexical structure

lassign(var_name, new_value)

The lexical elements of the Builtin lassign are shown in Table B.122.

Table B.122 – Builtin lassign

Parameter name	Type	Direction	Usage	Description
var_name	reference	I	m	specifies the name of the VARIABLE
new_value	int	I	m	specifies the new value of the variable
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE, if the assignment was successful, and FALSE, if the variable identifier was invalid.

B.123 Builtin lgetval

Purpose

The Builtin lgetval is specifically designed to be used in pre-read/write and post-read/write methods. This function returns the value of a VARIABLE as it was received from the connected device. Scaling operations may then be performed on the VARIABLE value before it is stored in the EDD application.

Lexical structure

lgetval(VIOD)

The lexical elements of the Builtin lgetval are shown in Table B.123.

Table B.123 – Builtin lgetval

Parameter name	Type	Direction	Usage	Description
<return>	int	O	m	specifies the value of the variable

B.124 Builtin ListDeleteElementAt

Purpose

Builtin ListDeleteElementAt will delete an element from a list. If the list is a member of a FILE, the specified element will be removed from the persistent store.

If a COUNT are explicitly declared on the specified LIST, any variables referenced by the COUNT attribute will not be updated. The method is responsible for updating the COUNT when elements are deleted from the list. If a COUNT are not explicitly declared, the COUNT attribute will be automatically updated.

If the index parameter is less than zero or greater than the number of elements in the list, an error will be returned. If the list parameter does not refer to a LIST instance, an error will be returned.

Lexical structure

ListDeleteElementAt(list, index)

The lexical element of the Builtin ListDeleteElementAt is shown in Table B.124.

Table B.124 – Builtin ListDeleteElementAt

Parameter name	Type	Direction	Usage	Description
list	reference	I	m	specifies the name of a LIST instance
index	int	I	m	specifies the index of the element to be deleted
<return>	long	O	m	specifies the return value of the Builtin

B.125 Builtin ListInsert

Purpose

Builtin ListInsert will insert an element to a list. If the list is a member of a FILE, the specified element will be added to the persistent store.

If COUNT and/or CAPACITY attributes are explicitly declared on the specified LIST, any variables referenced by these attributes will not be updated. The method is responsible for updating the COUNT and CAPACITY when elements are inserted in the list. If a COUNT and/or CAPACITY attributes are not explicitly declared, these attributes will be updated automatically.

If the index parameter is less than zero or greater than the capacity of the list, an error will be returned. If the list parameter does not refer to a LIST instance, an error will be returned.

Lexical structure

ListInsert(list, index, item)

The lexical element of the Builtin ListInsert is shown in Table B.125.

Table B.125 – Builtin ListInsert

Parameter name	Type	Direction	Usage	Description
list	reference	I	m	specifies the name of a LIST instance
index	int	I	m	specifies the index of the requested element
item	reference	I	m	specifies the name of the item to be inserted
<return>	long	O	m	specifies the return value of the Builtin

B.126 Builtin log**Purpose**

The Builtin log returns the natural logarithm of a floating point value.

Lexical structure

$\log(x)$

The lexical elements of the Builtin log are shown in Table B.126.

Table B.126 – Builtin log

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the natural logarithm

B.127 Builtin log10**Purpose**

The Builtin exp returns the base 10 logarithm of a floating point value.

Lexical structure

$\log_{10}(x)$

The lexical elements of the Builtin log10 are shown in Table B.127.

Table B.127 – Builtin log10

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the base 10 logarithm

B.128 Builtin log2**Purpose**

The Builtin log2 returns the base 2 logarithm of a floating-point value.

Lexical structure

`log2(x)`

The lexical elements of the Builtin `log2` are shown in Table B.128.

Table B.128 – Builtin `log2`

Parameter name	Type	Direction	Usage	Description
<code>x</code>	double	I	m	specifies a floating point value
<code><return></code>	double	O	m	returns the base 2 logarithm

B.129 Builtin `LOG_MESSAGE`

Purpose

Builtin `LOG_MESSAGE` inserts a message into a logging protocol.

Lexical structure

`LOG_MESSAGE(priority, message)`

The lexical elements of the Builtin `LOG_MESSAGE` are shown in Table B.129.

Table B.129 – Lexical elements of Builtin `LOG_MESSAGE`

Parameter name	Type	Direction	Usage	Description
<code>priority</code>	int	I	m	specifies the priority of the message. 0: error, 1: warning, 2: low priority message
<code>message</code>	char[]	I	m	specifies the message string to insert into the logging protocol
<code><return></code>	VOID	O	m	—

B.130 Builtin `long_value`

Purpose

The Builtin `long_value` will return the value of the specified device variable. The variable identifier shall be valid and of long type.

Lexical structure

`long_value(source_var_name)`

The lexical elements of the Builtin `long_value` are shown in Table B.130.

Table B.130 – Builtin `long_value`

Parameter name	Type	Direction	Usage	Description
<code>source_var_name</code>	reference	I	m	specifies the name of the variable
<code><return></code>	int	O	m	specifies the value of the variable

B.131 Builtin Isetval

Purpose

The Builtin Isetval will set the value of the device variable of long type for which a pre/post action has been designated to the specified value. It is specifically designed to be used in pre-read/write and post-read/write methods. This function will set the value of a device variable to the specified value.

Lexical structure

```
lsetval(value)
```

The lexical elements of the Builtin Isetval are shown in Table B.131.

Table B.131 – Builtin Isetval

Parameter name	Type	Direction	Usage	Description
value	int	I	m	specifies the new value of the VARIABLE
<return>	int	O	m	specifies the value of the variable

B.132 Builtin lvar_value

Purpose

The Builtin lvar_value will return the value of the specified device variable. The variable identifier shall be valid and of long type.

Lexical structure

```
lvar_value(source_var_name)
```

The lexical elements of the Builtin lvar_value are shown in Table B.132.

Table B.132 – Builtin lvar_value

Parameter name	Type	Direction	Usage	Description
source_var_name	reference	I	m	specifies the name of the VARIABLE
<return>	int	O	m	specifies the value of the variable

B.133 Builtin MEMBER_ID

Purpose

The Builtin MEMBER_ID can be used in methods to specify the member ID of a variable for calls to other Builtins. It returns the member ID of a BLOCK, PARAMETER_LIST, RECORD, REFERENCE_ARRAY, COLLECTION or VALUE_ARRAY.

Lexical structure

```
MEMBER_ID(name)
```

The lexical elements of the Builtin MEMBER_ID are shown in Table B.133.

Table B.133 – Builtin MEMBER_ID

Parameter name	Type	Direction	Usage	Description
name	reference	I	m	specifies the name of an item member
<return>	unsigned long	O	m	specifies the referenced member ID

B.134 Builtin MenuDisplay

Purpose

Builtin MenuDisplay will display a MENU instance to the user and a supplied list of options for the user to select from. Once a selection is made, the option position in the option list is returned. The position in the list is zero-based. For example, if the provided list contains two items and the first item is selected, this function will return a 0. If the second item is selected, a 1 is returned. The options are passed to the function as a single string with semicolon separators between the options.

Lexical structure

MenuDisplay(menu, options, selection)

If the EDD application is unable to display the menu, an error will be returned.

The lexical element of the Builtin MenuDisplay is shown in Table B.134.

Table B.134 – Builtin MenuDisplay

Parameter name	Type	Direction	Usage	Description
menu	reference	I	m	specifies the MENU to be displayed
options	char[]	I	m	specifies the option string, consisting of separated text, that defines a set of options presented to the user
selection	long*	O	m	specifies the zero-based selected option (see option parameter)
<return>	long	O	m	specifies the return value of the Builtin

B.135 Builtin method_abort

Purpose

The Builtin method_abort displays a message indicating that the method is aborting and waits for acknowledgment from the user.

After the acknowledgment, the abort methods in the abort method list are executed one after another from the front of the list (where the first abort method was added to the list) to the back of the list. The abort method list is maintained only for the execution of a single invocation of the method.

After the abort methods are executed, the method terminates and control is returned to the application.

If an abort method calls method_abort, the abort method list processing is immediately terminated.

Lexical structure

`method_abort(prompt)`

The lexical elements of the Builtin `method_abort` are shown in Table B.135.

Table B.135 – Builtin `method_abort`

Parameter name	Type	Direction	Usage	Description
<code>prompt</code>	<code>char[]</code>	I	m	specifies a message to be displayed
<code><return></code>	VOID	O	m	—

B.136 Builtin `ObjectReference`

Purpose

The Builtin `ObjectReference` returns unique id of an object. With the return object reference it is possible to have access to the variable and methods of other objects that may have different EDD.

NOTE For example, in an EDD of a remote I/O head station, it is possible to make a consistency check with the modules using `ObjectReference(CHILD)`.

Lexical structure

`ObjectReference(string)`

The lexical element of the Builtin `ObjectReference` is shown in Table B.136.

Table B.136 – Builtin `process_abort`

Parameter name	Type	Direction	Usage	Description
<code>string</code>	<code>char[]</code>	I	m	for the direction of navigation or the name of an object, see Table 148
<code><return></code>	<code>long</code>	O	m	specifies an ID which represent an object reference

B.137 Builtin `pow`

Purpose

The Builtin `pow` returns the power function of two floating-point values.

Lexical structure

`pow(x, y)`

The lexical elements of the Builtin `pow` are shown in Table B.137.

Table B.137 – Builtin `pow`

Parameter name	Type	Direction	Usage	Description
<code>x</code>	<code>double</code>	I	m	specifies the base
<code>y</code>	<code>double</code>	I	m	specifies the exponent
<code><return></code>	<code>double</code>	O	m	returns x raised to the y power

B.138 Builtin process_abort

Purpose

The Builtin process_abort will abort the current method, running any abort methods, which are in the abort method list. Unlike the Builtin abort, no message will be displayed when this function is executed. This Builtin function may not be run from inside an abort method.

Lexical structure

process_abort (VOID)

The lexical element of the Builtin process_abort is shown in Table B.138.

Table B.138 – Builtin process_abort

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.139 Builtin put_date

Purpose

The Builtin put_date stores the new un-scaled date value of the variable being processed.

The Builtin put_date can be called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action. The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a variable, which has one of these actions defined as an attribute, is written by an application, a user method, or an edit method.

Lexical structure

put_date (data, size)

The lexical elements of the Builtin put_date are shown in Table B.139.

Table B.139 – Builtin put_date

Parameter name	Type	Direction	Usage	Description
data	char[]	I	m	specifies the new value of the VARIABLE
size	long	I	m	specifies the length of the new value
<return>	long	O	m	is any of the return codes specified in Table B.235

B.140 Builtin put_date_value

Purpose

The Builtin put_date_value stores the provided value of the specified scaled variable into the variable table (caching EDD application) or device.

The parameters id and member id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the variable in question.

It puts the specified value of the date_and_time, time, or duration variable into the variable table or device.

The variable shall have a data type of char. The char string may contain up to eight chars and contain a variable of the following data types:

- DATA_AND_TIME
- TIME
- DURATION

The variable shall be valid in the variable table or device.

Lexical structure

```
put_date_value(id, member_id, data, size)
```

The lexical elements of the Builtin put_date_value are shown in Table B.140.

Table B.140 – Builtin put_date_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	o	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	o	specifies the member ID of the VARIABLE access
data	char[]	I	m	specifies the new value of the VARIABLE
size	long	I	m	specifies the length of the new value
<return>	long	O	m	is any of the return codes specified in Table B.235

B.141 Builtin put_double

Purpose

The Builtin put_double can be called only by methods associated with a pre-edit, post-edit post-read, or pre-write action.

The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a variable, which has one of these actions defined as an attribute, is written by an application, a user method, or an edit method.

Stores the new value of the double variable as it resides in the device (un-scaled). The Builtin put_double stores the new un-scaled double value of the variable being processed.

Lexical structure

```
put_double(value)
```

The lexical elements of the Builtin put_double are shown in Table B.141.

Table B.141 – Builtin put_double

Parameter name	Type	Direction	Usage	Description
value	double	I	m	specifies the new value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.142 Builtin put_double_value

Purpose

The Builtin put_double_value stores the provided value of the specified scaled variable into the variable table (caching EDD application) or device.

The parameters id and member_id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the variable in question.

Puts the specified value of a double variable into the variable table or device. The variable shall have a data type of double and be a valid variable in the variable table or device.

Lexical structure

```
put_double_value(id, member_id, value)
```

The lexical elements of the Builtin put_double_value are shown in Table B.142.

Table B.142 – Builtin put_double_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	o	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	o	specifies the member ID of the VARIABLE access
value	double	I	m	specifies the new value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.143 Builtin put_float

Purpose

The Builtin put_float stores the new unscaled floating-point value of the variable being operated on.

The Builtin put_float can be called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action. The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a variable, which has one of these actions defined as an attribute, is written by an application, a user method, or an edit method.

Lexical structure

```
put_float(value)
```

The lexical elements of the Builtin put_float are shown in Table B.143.

Table B.143 – Builtin put_float

Parameter name	Type	Direction	Usage	Description
value	double	I	m	specifies the new value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.144 Builtin put_float_value

Purpose

The Builtin put_float_value stores the provided value of the specified scaled variable in the variable table (caching EDD application) or the device.

The type of value is double to accommodate automatic promotion, which could occur whenever two floating-point values are multiplied.

The parameters id and member_id are specified using the Builtins ITEM_ID and MEMBER_ID Builtins and the name or reference of the variable in question. The variable shall have a data type of float and be a valid variable in the variable table or device.

Lexical structure

```
put_float_value(id, member_id, value)
```

The lexical elements of the Builtin put_float_value are shown in Table B.144.

Table B.144 – Builtin put_float_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	o	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	o	specifies the member ID of the VARIABLE access
value	double	I	m	specifies the new value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.145 Builtin PUT_MESSAGE

Purpose

The Builtin PUT_MESSAGE will display the specified message on the screen. Any embedded local variable references will be expanded to the variable's value (see Builtin put_message for Lexical Structure).

Embedded device variables are NOT supported in this function.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

```
PUT_MESSAGE(message)
```

The lexical elements of the Builtin PUT_MESSAGE are shown in Table B.145.

Table B.145 – Builtin PUT_MESSAGE

Parameter name	Type	Direction	Usage	Description
message	char[]	I	m	specifies a message to be displayed
<return>	VOID	—	—	—

B.146 Builtin put_message

Purpose

The Builtin put_message will display the specified message on the screen. Any embedded variable references will be expanded to the current value of the variable.

Any number of identifiers may be accessed by the string, but the array entries accessed shall contain valid variable identifiers. Local and device variables may be mixed in the display message.

Multiple languages are also supported in the put_message string. This is done by separating the different language strings by their language code identifier. The language code identifier is the vertical bar character, '|', followed immediately by that country's three-digit phone code. If the phone code is less than three digits in length, it should be padded on the left with zeros. For example, the following string is defined for both English and German.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

```
put_message(message, global_var_ids)
```

The lexical elements of the Builtin put_message are shown in Table B.146.

Table B.146 – Builtin put_message

Parameter name	Type	Direction	Usage	Description
message	char[]	I	m	specifies a message to be displayed
global_var_ids	array of int	I	m	specifies an array which contains unique identifiers of VARIABLE instance
<return>	VOID	—	—	—

B.147 Builtin put_signed

Purpose

The Builtin put_signed stores the new un-scaled signed value of the variable being operated on.

The Builtin put_signed can be called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action. The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a variable, which has one of these actions defined as an attribute, is written by an application, a user method or an edit method.

Lexical structure

```
put_signed(value)
```

The lexical elements of the Builtin put_signed are shown in Table B.147.

Table B.147 – Builtin put_signed

Parameter name	Type	Direction	Usage	Description
value	long	I	m	specifies the new value of the VARIABLE

Parameter name	Type	Direction	Usage	Description
<return>	long	O	m	is any of the return codes specified in Table B.235

B.148 Builtin put_signed_value

Purpose

The Builtin put_signed_value stores the new value of the specified scaled variable into the variable table (caching EDD application) or device.

The parameters id and member_id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the variable in question. Puts the specified value of a signed integer variable into the variable table or device.

The variable to be written shall have a long data type and be a valid variable in the variable table or device.

Lexical structure

```
put_signed_value(id, member_id, value)
```

The lexical elements of the Builtin put_signed_value are shown in Table B.148.

Table B.148 – Builtin put_signed_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	o	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	o	specifies the member ID of the VARIABLE access
value	long	I	m	specifies the new value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.149 Builtin put_string

Purpose

The Builtin put_string stores the new un-scaled string value of the variable being operated on.

The variable shall be valid in the variable table or device.

The variable shall have a char data type; a char string may contain device variables of the following data types:

- ASCII
- PASSWORD
- EUC (Extended Unit Code)
- BITSTRING

Device variables with data types of enumerated, bit_enumerated, and index shall be handled by the method because they are not supported by ANSI-C. The Builtin put_string can be called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action. The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a

variable, which has one of these actions defined as an attribute, is written by an application, a user method or an edit method.

Lexical structure

```
put_string(string, len)
```

The lexical elements of the Builtin `put_string` are shown in Table B.149.

Table B.149 – Builtin `put_string`

Parameter name	Type	Direction	Usage	Description
string	char[]	I	m	specifies the new value of the VARIABLE
len	long	I	m	specifies the length of the new value
<return>	long	O	m	is any of the return codes specified in Table B.235

B.150 Builtin `put_string_value`

Purpose

The Builtin `put_string_value` stores the provided value of the specified scaled variable into the variable table (caching EDD application) or device.

The parameters `id` and `member_id` are specified using the Builtins `ITEM_ID` and `MEMBER_ID` and the name or reference of the variable in question.

The variable shall be valid in the variable table or device. The variable shall have a data type of char; a char string may contain device variables of the following data types:

- ASCII
- PASSWORD
- EUC (Extended Unit Code)
- BITSTRING

Device variables with data types of enumerated, bit_enumerated, and index shall be handled by the method because they are not supported by ANSI-C.

Lexical structure

```
put_string_value(id, member_id, string, len)
```

The lexical elements of the Builtin `put_string_value` are shown in Table 150

Table B.150 – Builtin `put_string_value`

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	o	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	o	specifies the member ID of the VARIABLE access
string	char[]	I	m	specifies the new value of the VARIABLE
len	long	I	m	specifies the length of the new value
<return>	long	O	m	is any of the return codes specified in Table B.235

B.151 Builtin put_unsigned

Purpose

The Builtin put_unsigned stores the new unscaled unsigned value of the variable being operated on. The variable shall have a data type of unsigned and be a valid variable in the variable table or device. An unsigned value may contain device variables of the following data types:

- UNSIGNED
- ENUMERATED
- BIT_ENUMERATED
- INDEX

Device variables of these data types shall be handled by the method because they are not supported by ANSI-C. The Builtin put_unsigned should be called only by methods associated with a pre-edit, post-edit, post-read, or pre-write action. The methods associated with pre-edit, post-edit, post-read, or pre-write actions are run when a variable, which has one of these actions defined as an attribute, is written by an application or another method.

Lexical structure

put_unsigned(value)

The lexical elements of the Builtin put_unsigned are shown in Table B.151.

Table B.151 – Builtin put_unsigned

Parameter name	Type	Direction	Usage	Description
value	unsigned long	I	o	specifies the new value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.152 Builtin put_unsigned_value

Purpose

The Builtin put_unsigned_value stores the provided value of the specified scaled variable into the variable table (caching EDD application) or device.

The parameters id and member_id are specified using the Builtins ITEM_ID and MEMBER_ID and the name or reference of the variable in question.

The variable shall have a data type of unsigned and be a valid variable in the variable table or device. An unsigned value may contain device variables of the following data types:

- UNSIGNED
- ENUMERATED
- BIT_ENUMERATED
- INDEX

Device variables of these data types shall be handled by the method because they are not supported by ANSI-C. The variable shall be valid in the variable table or device.

Lexical structure

```
put_unsigned_value(id, member_id, value)
```

The lexical elements of the Builtin `put_unsigned_value` are shown in Table B.152.

Table B.152 – Builtin `put_unsigned_value`

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	o	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	o	specifies the member ID of the VARIABLE access
value	unsigned long	I	o	specifies the new value of the VARIABLE
<return>	long	O	m	is any of the return codes specified in Table B.235

B.153 Builtin READ_COMMAND

Purpose

Builtin `READ_COMMAND` allows reading variables with a given command.

Lexical structure

```
READ_COMMANDR(command_identifier)
```

The lexical elements of the Builtin `READ_COMMAND` are shown in Table B.153.

Table B.153 – Lexical elements of Builtin `READ_COMMAND`

Parameter name	Type	Direction	Usage	Description
command_identifier	reference	I	m	specifies the Identifier of the command to execute
<return>	VOID	—	—	—

B.154 Builtin read_value

Purpose

The Builtin `read_value` causes the value of the parameter to be read from the device.

The Builtin `read_value` does nothing in a non-caching EDD application because a non-caching EDD application does not have a variable table to update.

The Builtin `read_value` subjects to the same read and write timeouts as when an application is reading or writing a value. The parameters `id` and `member_id` are specified using the Builtins `ITEM_ID` and `MEMBER_ID` and the name or reference of the variable in question.

Lexical structure

```
read_value(id, member_id)
```

The lexical elements of the Builtin `read_value` are shown in Table B.154.

Table B.154 – Builtin read_value

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	o	specifies the item ID of the VARIABLE to access
member_id	unsigned long	I	o	specifies the member ID of the VARIABLE access
<return>	long	O	m	is any of the return codes specified in Table B.235

B.155 Builtin remove_abort_method (version A)**Purpose**

The Builtin `remove_abort_method` will remove a method from the abort method list, which is the list of methods to be executed if the current method is aborted. This Builtin will remove the first occurrence of the specified method in the list, starting with the first method added. If there are multiple occurrences of a specific method, only the first one is removed. Abort methods may not be removed during an abort method.

Lexical structure

```
remove_abort_method(abort_method_name)
```

The lexical elements of the Builtin `remove_abort_method` are shown in Table B.155.

Table B.155 – Builtin remove_abort_method

Parameter name	Type	Direction	Usage	Description
abort_method_name	reference	I	m	specifies the identifier of the method
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE, if the method was successfully removed from the list, and FALSE, if either the method was not in the list or if this function was run during an abort method.

B.156 Builtin remove_abort_method (version B)**Purpose**

The Builtin `remove_abort_method` moves a method from the abort method list. The abort method list contains the methods that are to be executed if the primary method aborts.

This Builtin removes the first occurrence of the method that it finds in the list, starting at the front of the list and moving to the back. The front of the list has the abort methods that were added first (FIFO). If the method occurs more than once in the list, the subsequent occurrences are not removed.

The `method_id` parameter is specified by the method writer using the Builtins `ITEM_ID` and the name of the method. Abort methods cannot call `remove_abort_method`.

Lexical structure

```
remove_abort_method(method_id)
```

The lexical elements of the Builtin `remove_abort_method` are shown in Table B.156.

Table B.156 – Builtin remove_abort_method

Parameter name	Type	Direction	Usage	Description
method_id	unsigned long	I	m	specifies the ID of the method
<return>	long	O	m	is any of the return codes specified in Table B.235

B.157 Builtin remove_all_abort_methods**Purpose**

The Builtin remove_all_abort_methods removes all methods from the abort method list. This Builtin is similar to the Builtin remove_abort_method or remove_abort_method except it removes all of the abort methods. An abort method cannot call remove_all_abort_methods.

Lexical structure

```
remove_all_abort_methods(VOID)
```

The lexical element of the Builtin remove_all_abort_methods is shown in Table B.157.

Table B.157 – Builtin remove_all_abort_methods

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.158 Builtin resolve_array_ref**Purpose**

The Builtin resolve_array_ref is used by method developers when parsing things such as item_array_name. The Builtin resolve_array_ref takes the item ID of the REFERENCE_ARRAY and an index and resolves them into the item ID of the REFERENCE_ARRAY element referenced. This Builtin can be used to fill arrays that are passed to display Bultins.

The parameters id and member_id are specified using the Bultins ITEM_ID and MEMBER_ID and the name or reference of the variable in question.

If a member of a REFERENCE_ARRAY has a conditional dependency (that is, the definition of an element is dependent on another value), the dependencies are taken into account during the resolution.

Lexical structure

```
resolve_array_ref(id, member_id)
```

The lexical elements of the Builtin resolve_array_ref are shown in Table B.158.

Table B.158 – Builtin resolve_array_ref

Parameter name	Type	Direction	Usage	Description
id	unsigned long	I	m	specifies the item ID of the REFERENCE_ARRAY
member_id	unsigned long	I	m	specifies the index of the requested REFERENCE_ARRAY element
<return>	unsigned long	O	m	specifies the item ID of the resolved reference, or zero for an error. In the case of an error, Builtin get_resolve_status may be used to access the actual error code

B.159 Builtin resolve_block_ref**Purpose**

The Builtin resolve_block_ref is used by method developers when parsing things such as BLOCK_A.name. The Builtin resolve_block_ref takes the member ID of a member of a block's characteristic record and returns the item ID for the resolved element.

The parameter member_id is specified using the MEMBER_ID Builtin and the name or reference of the VARIABLE in question. If a member of the characteristics record has a conditional dependency (that is, the definition of an element is dependent on another value), the dependencies are taken into account during the resolution.

Resolves the member_id reference generated for a BLOCK_A element into an item ID, which is then used in Builtin calls.

Lexical structure

```
resolve_block_ref(member_id)
```

The lexical elements of the Builtin resolve_block_ref are shown in Table B.159.

Table B.159 – Builtin resolve_block_ref

Parameter name	Type	Direction	Usage	Description
member_id	unsigned long	I	m	specifies the member ID of BLOCK_A
<return>	unsigned long	O	m	specifies the item ID of the resolved reference, or zero for an error. In the case of an error, Builtin get_resolve_status may be used to access the actual error code

B.160 Builtin resolve_param_list_ref**Purpose**

The Builtin resolve_param_list_ref is used by method developers when parsing things such as PARAM_LIST name. It takes the member ID of a PARAMETER_LIST element and returns the item ID of the completely resolved element.

The parameter member_id is specified using the MEMBER_ID Builtin and the name or reference of the variable in question. If a member of the parameter list has a conditional dependency—the definition of an element is dependent on another value – the dependencies are taken into account during the resolution.

The Builtin `resolve_param_list_ref` resolves the `member_id` reference generated for a `PARAMETER_LIST` element into an item ID, which is then used in Builtin calls.

Lexical structure

`resolve_param_list_ref(member_id)`

The lexical elements of the Builtin `resolve_param_list_ref` are shown in Table B.160.

Table B.160 – Builtin `resolve_param_list_ref`

Parameter name	Type	Direction	Usage	Description
<code>member_id</code>	unsigned long	I	m	specifies the member ID of the parameter list
<code><return></code>	unsigned long	O	m	specifies the item ID of the resolved reference, or zero for an error. In the case of an error, Builtin <code>get_resolve_status</code> may be used to access the actual error code

B.161 Builtin `resolve_param_ref`

Purpose

The Builtin `resolve_param_ref` is used by method developers when parsing things such as `PARAM.name`. It takes the member ID of a `PARAMETER` element and returns the item ID of the completely resolved element.

The parameter `member_id` is specified using the `MEMBER_ID` Builtin and the name or reference of the variable in question. It resolves the `member_id` reference generated for a `PARAMETER` element into an item ID, which is then used in Builtin calls.

Lexical structure

`resolve_param_ref(member_id)`

The lexical elements of the Builtin `resolve_param_ref` are shown in Table B.161.

Table B.161 – Builtin `resolve_parm_ref`

Parameter name	Type	Direction	Usage	Description
<code>member_id</code>	unsigned long	I	m	specifies the member ID of the parameter
<code><return></code>	unsigned long	O	m	specifies the item ID of the resolved reference, or zero for an error. In the case of an error, Builtin <code>get_resolve_status</code> may be used to access the actual error code

B.162 Builtin `resolve_record_ref`

Purpose

The Builtin `resolve_record_ref` is used by method developers when parsing things such as `record_name.member_name`. It takes the item ID of a record and the member ID of a record element or member and returns the item ID of the completely resolved element.

If a member of a record has a conditional dependency (that is, the definition of an element is dependent on another value), the dependencies are taken into account during the resolution.

The parameters `id` and `member_id` are specified using the Builtins `ITEM_ID` and `MEMBER_ID` and the name or reference of the variable in question.

Resolves the `member_id` reference generated for a `RECORD` element into an item ID, which is then used in Builtin calls.

Lexical structure

```
resolve_record_ref(id, member_id)
```

The lexical elements of the Builtin `resolve_record_ref` are shown in Table B.162.

Table B.162 – Builtin `resolve_record_ref`

Parameter name	Type	Direction	Usage	Description
<code>id</code>	unsigned long	I	m	specifies the item ID of the <code>RECORD</code> being resolved
<code>member_id</code>	unsigned long	I	m	specifies the member ID of the record member
<code><return></code>	unsigned long	O	m	specifies the item ID of the resolved reference, or zero for an error. In the case of an error, Builtin <code>get_resolve_status</code> may be used to access the actual error code

B.163 Builtin `retry_on_all_comm_errors`

Purpose

The Builtin `retry_on_all_comm_errors` causes the last request made by a method to be tried again after any communications error. A request will be retried three times before it fails and an error return is returned.

This Builtin frees a method from having to explicitly retry a request after any communications errors.

The communication error codes are defined in the standard profiled and their related strings are defined in the text dictionary.

NOTE A communications error of zero is not considered a communications error.

Lexical structure

```
retry_on_all_comm_errors(VOID)
```

The lexical element of the Builtin `retry_on_all_comm_errors` is shown in Table B.163.

Table B.163 – Builtin `retry_on_all_comm_errors`

Parameter name	Type	Direction	Usage	Description
<code><return></code>	VOID	—	—	—

B.164 Builtin RETRY_ON_ALL_COMM_STATUS

Purpose

The Builtin RETRY_ON_ALL_COMM_STATUS will set all of the bits in the comm status retry mask. This will cause the system to retry the current command if the device returns any comm status value. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is set.

The retry and abort masks are reset to their default values at the start of each method. so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks, and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_ALL_COMM_STATUS (VOID)

The lexical element of the Builtin RETRY_ON_ALL_COMM_STATUS is shown in Table B.164.

Table B.164 – Builtin RETRY_ON_ALL_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.165 Builtin RETRY_ON_ALL_DEVICE_STATUS

Purpose

The Builtin RETRY_ON_ALL_DEVICE_STATUS will set all of the bits in the device status retry mask. This will cause the system to retry the current command if the device returns any device status value. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_ALL_DEVICE_STATUS (VOID)

The lexical element of the Builtin RETRY_ON_ALL_DEVICE_STATUS is shown in Table B.165.

Table B.165 – Builtin RETRY_ON_ALL_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.166 Builtin RETRY_ON_ALL_RESPONSE_CODES

Purpose

The Builtin RETRY_ON_ALL_RESPONSE_CODES will set all of the bits in the response code retry mask. This will cause the System to retry the current command if the device returns any code value.

The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks, and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_ALL_RESPONSE_CODES (VOID)

The lexical element of the Builtin RETRY_ON_ALL_RESPONSE_CODES is shown in Table B.166.

Table B.166 – Builtin RETRY_ON_ALL_RESPONSE_CODES

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.167 Builtin retry_on_all_response_codes

Purpose

The Builtin retry_on_all_response_codes causes the last request made by a method to be retried after any response code. A request will be retried three times before it fails and an error return is returned. This is the default action by the Builtins when a method is just started.

This Builtin frees a method from having to explicitly retry a request after any response code.

A response code is an integer value representing an application-specific error condition.

NOTE A response code value of zero is not considered an error response code. Each response code and response code type is defined in the EDD.

Lexical structure

retry_on_all_response_codes (void)

The lexical element of the Builtin retry_on_all_response_codes is shown in Table B.167.

Table B.167 – Builtin retry_on_all_response_codes

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.168 Builtin RETRY_ON_COMM_ERROR

Purpose

The Builtin RETRY_ON_COMM_ERROR will set the no comm error mask such that the current command will be retried if a comm error is found while sending the command.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_COMM_ERROR(VOID)

The lexical element of the Builtin RETRY_ON_COMM_ERROR is shown in Table B.168.

Table B.168 – Builtin RETRY_ON_COMM_ERROR

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	

B.169 Builtin retry_on_comm_error

Purpose

The Builtin retry_on_comm_error causes the last request made by a method to be retried after a specified communications error. A request will be retried three times before it fails and an error return is returned.

This Builtin frees a method from having to explicitly retry a request after a given communications error is received.

The communications error codes are defined by the consortia. Their related strings may be defined in a text dictionary. A communications error of zero is not considered a communications error.

Lexical structure

retry_on_comm_error(error)

The lexical elements of the Builtin retry_on_comm_error are shown in Table B.169.

Table B.169 – Builtin retry_on_comm_error

Parameter name	Type	Direction	Usage	Description
error	unsigned long	I	m	specifies the communication error
<return>	long	O	m	is any of the return codes specified in Table B.235

B.170 Builtin RETRY_ON_COMM_STATUS

Purpose

The Builtin RETRY_ON_COMM_STATUS will set the correct bit(s) in the comm status retry mask such that the specified comm status value will cause the current command to be retried. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 1.

The retry and abort masks are result to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_method function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_COMM_STATUS(comm_status)

The lexical elements of the Builtin RETRY_ON_COMM_STATUS are shown in Table B.170.

Table B.170 – Builtin RETRY_ON_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
comm_status	int	l	m	specifies the new comm status retry mask
<return>	VOID	—	—	—

B.171 Builtin RETRY_ON_DEVICE_STATUS

Purpose

The Builtin RETRY_ON_DEVICE_STATUS will set the correct bit(s) in the device status retry mask such that the specified device status value will cause the current command to be retried. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_DEVICE_STATUS(device_status)

The lexical elements of the Builtin RETRY_ON_DEVICE_STATUS are shown in Table B.171.

Table B.171 – Builtin RETRY_ON_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
device_status	int	l	m	specifies the new comm status retry mask
<return>	VOID	—	—	—

B.172 Builtin RETRY_ON_NO_DEVICE

Purpose

The Builtin RETRY_ON_NO_DEVICE will set the no device mask such that the current command will be retried if no device is found while sending a command.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_NO_DEVICE (VOID)

The lexical element of the Builtin RETRY_ON_NO_DEVICE is shown in Table B.172.

Table B.172 – Builtin RETRY_ON_NO_DEVICE

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.173 Builtin RETRY_ON_RESPONSE_CODE

Purpose

The Builtin RETRY_ON_RESPONSE_CODE will set the correct bit(s) in the response code retry mask such that the specified response code value will cause the current command to be retried. The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of available masks and their default values.

The mask affected by this function is used by the Builtins send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status and display.

Lexical structure

RETRY_ON_RESPONSE_CODE (response_code)

The lexical elements of the Builtin RETRY_ON_RESPONSE_CODE are shown in Table B.173.

Table B.173 – Builtin RETRY_ON_RESPONSE_CODE

Parameter name	Type	Direction	Usage	Description
response_code	int	I	m	specifies the new comm status retry mask
<return>	VOID	—	—	—

B.174 Builtin retry_on_response_code**Purpose**

The Builtin `retry_on_response_code` causes the last request made by a method to be retried three times after a specific response code is received.

This Builtin frees a method from having to explicitly retry a request after a given response code is received.

A response code is a value representing an application-specific error condition.

NOTE A response code value of zero is not considered an error response code. Each response code and response code type is defined in the EDD.

Retries the method's last request upon receiving the specified response code.

Lexical structure

```
retry_on_response_code (code)
```

The lexical elements of the Builtin `retry_on_response_code` are shown in Table B.174.

Table B.174 – Builtin retry_on_response_code

Parameter name	Type	Direction	Usage	Description
code	unsigned long	I	m	specifies the response code being resolved
<return>	long	O	m	is any of the return codes specified in Table B.235

B.175 Builtin round**Purpose**

The Builtin `round` rounds a floating point value to the nearest integer.

Lexical structure

```
round (x)
```

The lexical elements of the Builtin `round` are shown in Table B.175.

Table B.175 – Builtin round

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the rounded value

B.176 Builtin rspcode_string

Purpose

The Builtin rspcode_string will return the response code string for the specified command and response code. If the string is longer than the maximum length defined in response_string_length, the string is truncated. The response code specified shall be valid for the indicated command.

Lexical structure

```
rspcode_string(cmd_number, response_code, response_string,
               response_string_length)
```

The lexical elements of the Builtin rspcode_string are shown in Table B.176.

Table B.176 – Builtin rspcode_string

Parameter name	Type	Direction	Usage	Description
cmd_number	int	I	m	specifies the command number
response_code	int	I	m	specifies the response code
response_string	char*	O	m	specifies the buffer for the response code string
response_string_length	int	I		specifies the maximum length of the string buffer
<return>	VOID	—	—	—

B.177 Builtin save_on_exit

Purpose

The Builtin save_on_exit sets the termination action to save any changes made by the method to variables in the variable table not already sent to the device. The values in the device are not affected. As long as the application has not changed the values, the values are saved as if written by an application.

When a method modifies a variable's value, it is really just changing a resident copy of that value in the EDD application. This change is effective only for the life of the method. To make the values permanent in the variable table, the method shall save the values before exiting.

All values that have been changed, but not sent to the device, are handled at the termination of the method by either sending them to the device, discarding them, or saving the value changes in a way that survives the end of the method (but does not affect the device). Saved values can be used by other methods.

If a method does not call Builtin discard_on_exit, save_on_exit, or send_on_exit prior to exiting, any values in the Builtin are discarded as if discard_on_exit was called.

If the method aborts, then the Builtin discard_on_exit option is enforced even if Builtin save_on_exit were called before the method aborted.

Lexical structure

```
save_on_exit(VOID)
```

The lexical element of the Builtin save_on_exit is shown in Table B.177.

Table B.177 – Builtin save_on_exit

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.178 Builtin save_values**Purpose**

The Builtin save_values will cause the values of all device variables modified during a method session to remain permanent after the method has been exited.

Device variables that are modified, but are not sent to the connected device, will be restored to their previous value if this function is not called.

Lexical structure

save_values(VOID)

The lexical element of the Builtin save_values is shown in Table B.178.

Table B.178 – Builtin save_values

Parameter name	Type	Direction	Usage	Description
<return>	int	O	m	specifies the return value of the Builtin

B.179 Builtin SELECT_FROM_LIST**Purpose**

The Builtin SELECT_FROM_LIST has the same functionality as Builtin SELECT_FROM_LIST, except that device variables are not allowed in the prompt string. There shall be at least two options in the select list.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

SELECT_FROM_LIST(prompt, option_list)

The lexical elements of the Builtin SELECT_FROM_LIST are shown in Table B.179.

Table B.179 – Builtin SELECT_FROM_LIST

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
option_list	char[]	I	m	specifies the option string, consisting of separated text, that defines the menu items
<return>	int	O	m	specifies the result of the selection as position (zero based)

B.180 Builtin select_from_list

Purpose

The Builtin select_from_list will display a prompt and a supplied list of options for the user to select from. Once a selection is made, the position of that option in the option list is returned. The position in the list is zero-based. For example, if the list provided contains two items and if the first item is selected, this function will return a 0. If the second item is selected, a 1 is returned.

The prompt and/or list may also contain embedded local and/or device variable values (see put_message for the Lexical Structure).

The options are passed to the function as a single string with semicolon separators between the options. There shall be at least two options in the select list.

NOTE 1 Previously displayed messages are not guaranteed to remain on the screen.

NOTE 2 Some EDD applications will clear the display upon each call to this function.

Lexical structure

select_from_list(prompt, global_var_ids, option_list)

The lexical elements of the Builtin SELECT_FROM_LIST are shown in Table B.180.

Table B.180 – Builtin select_from_list

Parameter name	Type	Direction	Usage	Description
prompt	char[]	l	m	specifies a message to be displayed
global_var_ids	array of int	l	m	specifies an array which contains unique identifiers of VARIABLE instance
option_list	char[]	l	m	specifies the option string, consisting of separated text, that defines the menu items
<return>	int	o	m	specifies the result of the selection as position (zero based)

B.181 Builtin select_from_menu

Purpose

The Builtin select_from_menu displays a list of options or menu items for the user to select from and retrieves the selection number. When the user has made a choice it is returned to the calling method. Selection numbers start at 1 and range up to the number of options specified by the parameter options. Variable values may be embedded in the prompt message string with formatting information. The option string defines the menu from which the user selects and is a string consisting of the selection options; each option is separated from the other options with a semi-colon. Each option may have language codes embedded in it.

The application displays the strings in accordance with the following guidelines:

- the menu items appear on the screen in the same order as listed in the string;
- the prompt string appears at the top of the menu display on the screen;
- the prompt string is not selectable;
- the menu is presented vertically.

Lexical structure

```
select_from_menu(prompt, ids, indices, id_count, options, selection)
```

The lexical elements of the Builtin select_from_menu are shown in Table B.181.

Table B.181 – Builtin select_from_menu

Parameter name	Type	Direction	Usage	Description
prompt	char[]	I	m	specifies a message to be displayed
ids	array of unsigned long	I	m	specifies the item IDs of the VARIABLES used in the prompt message
indices	array of unsigned long	I	m	specifies the member IDs or indices of the VARIABLES used in the prompt message
id_count	long	I	m	specifies the number of REFERENCE_ARRAY elements in the REFERENCE_ARRAY for ids and indices. Set to zero if the REFERENCE_ARRAY for ids and indices are zero
options	char[]	I	m	specifies the option string, consisting of separated text, that defines the menu items
selection	long	O	m	specifies the result of the selection as position (zero based)
<return>	long	O	m	is any of the return codes specified in Table B.235

B.182 Builtin send

Purpose

The Builtin send will send the specified command and return the response code, communications status, and command status respectively in the cmd_status array.

The octets in the cmd_status array are logically AND with the appropriate abort masks to determine if the method should be aborted. If there is a match, the method is aborted and the abort methods (if any were designated) are run. If there is no match for the abort masks, the same data is AND with the appropriate retry masks. If there is a match, the command will be retried. This continues until either the command succeeds, an abort occurs, or the maximum number of retries is reached.

The status octets for the command are returned in the cmd_status parameter, a pointer to a 3-octet array, which is allocated in the method making the call.

Lexical structure

```
send(cmd_number, cmd_status)
```

The lexical elements of the Builtin send are shown in Table B.182.

Table B.182 – Builtin send

Parameter name	Type	Direction	Usage	Description
cmd_number	int	I	m	specifies the number of command to send
cmd_status	char*	O	m	specifies the status octets of the send command
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be BI_SUCESS, if the command was successful, BI_COMM_ERROR, if an error occurred sending the command, or BI_NO_DEVICE, if no device was found.

B.183 Builtin send_all_values

Purpose

The Builtin send_all_values sends all of the values in the variable table that the method modified but has not previously sent to the device. This action changes the device values to match the values set by the method.

This Builtin does nothing for a non-caching EDD application because it does not have a variable table. Updates the device with all of the values in the variable table that the method has changed.

Lexical structure

send_all_values (VOID)

The lexical element of the Builtin send_all_values is shown in Table B.183.

Table B.183 – Builtin send_all_values

Parameter name	Type	Direction	Usage	Description
<return>	long	O	m	is any of the return codes specified in Table B.235

B.184 Builtin send_command

Purpose

The Builtin send_command will send the specified command. If the status bit indicating that more data is available is returned (status class MORE), command 48 is automatically issued. The status/data octets from the original command and command 48 (if it was sent) are then logically AND with the appropriate abort masks to determine if the method should be aborted. If there is a match, the method is aborted and the abort methods (if any were designated) are run. If there is no match for the abort masks, the same data is AND with the appropriate retry masks. If there is a match, the appropriate command will be retried (either the specified command, or command 48). This continues until either the command succeeds, an abort occurs, or the maximum number of retries is reached. No status or data octets are returned with this function.

Lexical structure

send_command (cmd_number)

The lexical elements of the Builtin send_command are shown in Table B.184.

Table B.184 – Builtin send_command

Parameter name	Type	Direction	Usage	Description
cmd_number	int	I	m	specifies the number of command to send
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be BI_SUCESS, if the command was successful, BI_COMM_ERROR, if an error occurred sending the command, or BI_NO_DEVICE, if no device was found.

B.185 Builtin send_command_trans

Purpose

The Builtin send_command_trans sends the command with the specified transaction to the device. This function is to be used to send commands that have been defined with multiple transactions.

The Builtin send_command_trans will send the specified command. If the status bit indicating that more data is available is returned (status class MORE), command 48 is automatically issued. The status/data octets from the original command and command 48 (if it was sent) are then logically ANDed with the appropriate abort masks to determine if the method should be aborted. If there is a match, the method for the abort masks, the same data is ANDed with the appropriate retry masks. If there is a match, the appropriate command will be retried (either the specified command, or command 48). This continues until either the command succeeds, an abort occurs, or the maximum number of retries is reached. No status or data octets are returned with this command.

Lexical structure

```
send_command_trans(cmd_number, transaction)
```

The lexical elements of the Builtin send_command_trans are shown in Table B.185.

Table B.185 – Builtin send_command_trans

Parameter Name	Type	Direction	Usage	Description
cmd_number	int	I	m	specifies the number of command to send
transaction	int	I	m	specifies the number of the TRANSACTION
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be BI_SUCESS if the command was successful, BI_COMM_ERROR if an error occurred sending the command, or BI_NO_DEVICE if no device was found.

B.186 Builtin send_on_exit

Purpose

The Builtin send_on_exit sets the termination action to send any changes made by the method to variables stored in the variable table of the EDD application, which have not already been sent to the device. This Builtin does nothing in a non-caching EDD application, because a non-caching EDD application does not have a variable table.

When a method modifies a variable value, it changes a copy of that value. This change is effective only for the life of the method. In order to make the change in the device, the method shall send the value(s) to the device or call this Builtin before the method exits.

All values that are changed, but not sent to the device, are handled at the termination of the method by either sending them to the device, discarding them, or saving the value changes in a way that survives the end of the method. The changed values in the variable table are sent after the method terminates.

If a method does not call Builtin discard_on_exit, or Builtin send_on_exit prior to the exiting, any values changed by the Builtin are discarded as if Builtin discard_on_exit was called. If the method aborts then the Builtin discard_on_exit option is enforced, even if Builtin save_on_exit, or Builtin send_on_exit were called before the method aborted.

Lexical structure

`send_on_exit(VOID)`

The lexical element of the Builtin `send_on_exit` is shown in Table B.186.

Table B.186 – Builtin `send_on_exit`

Parameter Name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.187 Builtin `send_trans`

Purpose

The Builtin `send_trans` sends a command with the specified transaction to the device. This function is to be used to send command that have been defined with multiple transactions.

The Builtin `send_trans` will send the specified command and return the response code, communications status, and command status respectively in the `cmd_status` array. These octets are logically ANDed with the appropriate abort masks to determine if the method should be aborted. If there is a match, the method is aborted and the abort methods (if any were designated) are run. If there is a match, the command will be retried. This continues until either the command succeeds, an abort occurs, or the maximum number of retries is reached.

The status octets for the command are returned in the `cmd_status` parameter, a 3-octet array, allocated in the method making the call.

Lexical structure

`send_trans(cmd_number, transaction, cmd_status)`

The lexical elements of the Builtin `send_trans` are shown in Table B.187.

Table B.187 – Builtin `send_trans`

Parameter name	Type	Direction	Usage	Description
<code>cmd_number</code>	int	I	m	specifies the number of command to send
<code>transaction</code>	int	I	m	specifies the number of the TRANSACTION
<code>cmd_status</code>	char*	O	m	specifies the status octets of the send command
<return>	int	O	m	specifies a status if the command was successfully, if an error occurred or no device was found

NOTE The symbolic name for the return value should be `BI_SUCESS`, if the command was successful, `BI_COMM_ERROR`, if an error occurred sending the command, or `BI_NO_DEVICE`, if no device was found.

B.188 Builtin `send_value`

Purpose

The Builtin `send_value` causes the value of the specked variable in the variable table to be sent to the device, thus updating the device value. This Builtin does nothing in a non-caching EDD application, because a non-caching EDD application does not have a variable table. Some values are `READ_ONLY` in the device and cannot be written. Attempting to do so will cause a Builtin error to be returned.

The parameters `id` and `member id` are specified using the Builtins `ITEM_ID` and `MEMBER_ID` and the name or reference of the variable in question.

Lexical structure

`send_value(id, member_id)`

The lexical elements of the Builtin `send_value` are shown in Table B.188.

Table B.188 – Builtin `send_value`

Parameter name	Type	Direction	Usage	Description
<code>id</code>	unsigned long	I	m	specifies the item ID of the VARIABLE to read
<code>member_id</code>	unsigned long	I	m	specifies the member ID of the VARIABLE to read
<code><return></code>	long	O	m	is any of the return codes specified in Table B.235

B.189 Builtin SET_NUMBER_OF_RETRIES

Purpose

The Builtin `SET_NUMBER_OF_RETRIES` sets up the number of times a command will be retried due to the appropriate retry masks. This value is defaulted to 0 at the beginning of each method.

Lexical structure

`SET_NUMBER_OF_RETRIES(n)`

The lexical elements of the Builtin `SET_NUMBER_OF_RETRIES` are shown in Table B.189.

Table B.189 – Builtin `SET_NUMBER_OF_RETRIES`

Parameter name	Type	Direction	Usage	Description
<code>n</code>	int	I	m	specifies the number of retries of a command
<code><return></code>	int	O	m	specifies the return value of the Builtin

B.190 Builtin sin

Purpose

The Builtin `sin` returns the sine a floating point value.

Lexical structure

`sin(x)`

The lexical elements of the Builtin `sin` are shown in Table B.190.

Table B.190 – Builtin `sin`

Parameter name	Type	Direction	Usage	Description
<code>x</code>	double	I	m	specifies a floating-point value
<code><return></code>	double	O	m	returns the sine value

B.191 Builtin sinh

Purpose

The Builtin sinh returns the hyperbolic sine of a floating point value.

Lexical structure

`sinh(x)`

The lexical elements of the Builtin sinh are shown in Table B.191.

Table B.191 – Builtin sinh

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the hyperbolic sine value

B.192 Builtin sqrt

Purpose

The Builtin sqrt returns the square root of a floating-point value.

Lexical structure

`sqrt(x)`

The lexical elements of the Builtin sqrt are shown in Table B.192.

Table B.192 – Builtin sqrt

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the square root

B.193 Builtin strcmp

Purpose

The Builtin strcmp compare two strings.

Lexical structure

`strcmp(string1, string2)`

The lexical elements of the Builtin strcmp are shown in Table B.193.

Table B.193 – Builtin strcmp

Parameter name	Type	Direction	Usage	Description
string1	DD_STRING	I	m	specifies the first string
string2	DD_STRING	I	m	specifies the second string
<return>	int	O	m	returns zero if string1 and string2 are the same, returns a negative value if string1 is less than string2, returns a positive value if string1 is greater than string2

B.194 Builtin strlen

Purpose

The Builtin strlen returns the length of a string.

Lexical structure

```
strlen(string)
```

The lexical elements of the Builtin strlen are shown in Table B.194.

Table B.194 – Builtin strlen

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the string whose length is to be returned
<return>	int	O	m	returns the length of the string

B.195 Builtin strlwr

Purpose

The Builtin strlwr returns a lower-case version of string.

Lexical structure

```
strlwr(string)
```

The lexical elements of the Builtin strlwr are shown in Table B.195.

Table B.195 – Builtin strlwr

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the string to be converted
<return>	DD_STRING	O	m	returns a lower-case version of the string

B.196 Builtin strmid

Purpose

The Builtin strmid returns a substring of a string.

Lexical structure

```
strmid(string, position, length)
```

The lexical elements of the Builtin strmid are shown in Table B.196.

Table B.196 – Builtin strmid

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the input string
position	int	I	m	specifies the first character to be extracted from the input string
length	int	I	m	specifies the number of characters to be extracted from the input string
<return>	DD_STRING	O	m	returns the specified substring

B.197 Builtin strstr

Purpose

The Builtin strstr returns a sub-string of a string.

Lexical structure

strstr(string, substring)

The lexical elements of the Builtin strstr are shown in Table B.197

Table B.197 – Builtin strstr

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the string to be searched
substring	DD_STRING	I	m	specifies the search string
<return>	DD_STRING	O	m	returns the first occurrence of s1 in s2, or NULL if the string cannot be found

B.198 Builtin strtrim

Purpose

The Builtin strtrim trims whitespace from a string.

Lexical structure

strtrim(string)

The lexical elements of the Builtin strtrim are shown in Table B.198.

Table B.198 – Builtin strtrim

Parameter name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the string to be trimmed
<return>	DD_STRING	O	m	returns a string with leading and trailing whitespace removed

B.199 Builtinstrupr

Purpose

The Builtinstrupr returns an upper-case version of string.

Lexical structure

`strupr(string)`

The lexical elements of the Builtinstrupr are shown in Table B.199.

Table B.199 – Builtinstrupr

Parameter Name	Type	Direction	Usage	Description
string	DD_STRING	I	m	specifies the string to be converted
<return>	DD_STRING	O	m	returns an upper-case version of the string

B.200 Builtin tan

Purpose

The Builtin tan returns the tangent of a floating point value.

Lexical structure

`tan(x)`

The lexical elements of the Builtin tan are shown in Table B.200.

Table B.200 – Builtin tan

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the tangent

B.201 Builtin tanh

Purpose

The Builtin tanh returns the hyperbolic tangent of a floating point value.

Lexical structure

`tanh(x)`

The lexical elements of the Builtin tanh are shown in Table B.201.

Table B.201 – Builtin tanh

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating point value
<return>	double	O	m	returns the hyperbolic tangent

B.202 Builtin To_Date_and_Time

Purpose

The Builtin To_Date_and_Time calculates the number of days and fractional days from a given date since 1 January 1900.

NOTE The return value can be assigned to a DATE_AND_TIME variable or can be compared with it.

Lexical structure

To_Date_and_Time(days, hour, minute, second, millisecond)

Table B.202 – Builtin To_Date_and_Time

Parameter name	Type	Direction	Usage	Description
days	int	I	m	specifies the number of days
hour	int	I	m	specifies the hours
minute	int	I	m	specifies the minutes
second	int	I	m	specifies the seconds
millisecond	int	I	m	specifies the milliseconds
<return>	double	O	m	returns the number of days and fractional days. The EDD application shall support an implicit typecast between double float and DATE_AND_TIME

B.203 Builtin trunc

Purpose

The Builtin trunc truncates a floating-point value.

Lexical structure

trunc(x)

The lexical elements of the Builtin trunc are shown in Table B.203.

Table B.203 – Builtin trunc

Parameter name	Type	Direction	Usage	Description
x	double	I	m	specifies a floating-point value
<return>	double	O	m	returns the truncated value

B.204 Builtin VARID

Purpose

The Builtin VARID will return the numeric identifier for the variable specified. A valid variable name shall be provided. Each variable in the device description is assigned a unique numeric identifier. This function is to be used when the identifier of a variable either needs to be stored in a temporary buffer, or needs to be sent as a parameter to another Builtin.

Lexical structure

VARID(variable_name)

The lexical elements of the Builtin VARID are shown in Table B.204

Table B.204 – Builtin VARID

Parameter name	Type	Direction	Usage	Description
variable_name	reference	I	m	specifies the identifier of the VARIABLE
<return>	int	O	m	specifies the variable identifier

B.205 Builtin vassign

Purpose

The Builtin vassign will assign the value of the source variable to the destination variable. Both variables shall be valid and they shall have the same type.

Lexical structure

vassign(dest_var, source_var)

The lexical elements of the Builtin vassign are shown in Table B.205.

Table B.205 – Builtin vassign

Parameter name	Type	Direction	Usage	Description
dest_var	reference	I	m	specifies the identifier of the destination VARIABLE
source_var	reference	I	m	specifies the identifier of the source VARIABLE
<return>	int	O	m	specifies the return value of the Builtin

NOTE The symbolic name for the return value should be TRUE if the assignment was successful, and FALSE if the variable identifier was invalid.

B.206 Builtin WRITE_COMMAND

Purpose

Builtin WRITE_COMMAND allows to write variables with a given command.

Lexical structure

WRITE_COMMAND(command_identifier)

The lexical elements of the Builtin WRITE_COMMAND are shown in Table B.206.

Table B.206 – Builtin WRITE_COMMAND

Parameter name	Type	Direction	Usage	Description
command_identifier	reference	I	m	specifies the Identifier of the COMMAND to execute
<return>	long	O	m	specifies the return value of the Builtin

B.207 Builtin XMTR_ABORT_ON_ALL_COMM_STATUS

Purpose

The Builtin XMTR_ABORT_ON_ALL_COMM_STATUS will set all of the bits in the command 48 comm status abort mask. This will cause the system to abort the current method if the device returns any command 48 comm status value. Comm status is defined to be the first octet returned in a transaction, when bit 7 of this octet is set.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_ALL_COMM_STATUS (VOID)

The lexical element of the Builtin XMTR_ABORT_ON_ALL_COMM_STATUS is shown in Table B.207.

Table B.207 – Builtin XMTR_ABORT_ON_ALL_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.208 Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS

Purpose

The Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS will set all of the bits in the command 48 device status abort mask. This will cause the system to abort the current method if the device returns any command 48 device status value. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_ALL_DEVICE_STATUS (VOID)

The lexical element of the Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS is shown in Table B.208.

Table B.208 – Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.209 Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES**Purpose**

The Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES will set all of the bits in the command 48 response code abort mask. This will cause the system to abort the current method if the device returns any command 48 response code value.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_ALL_RESPONSE_CODES(VOID)

The lexical element of the Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES is shown in Table B.209.

Table B.209 – Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.210 Builtin XMTR_ABORT_ON_COMM_ERROR**Purpose**

The Builtin XMTR_ABORT_ON_COMM_ERROR will set the command 48 no comm error mask such that the method will be aborted if a comm error is found while sending command 48.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_COMM_ERROR(VOID)

The lexical element of the Builtin XMTR_ABORT_ON_COMM_ERROR is shown in Table B.210.

Table B.210 – Builtin XMTR_ABORT_ON_COMM_ERROR

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.211 Builtin XMTR_ABORT_ON_COMM_STATUS

Purpose

The Builtin XMTR_ABORT_ON_COMM_STATUS will set the correct bit(s) in the command 48 comm status abort mask such that the specified command 48 comm status value will cause the method to abort. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 1. The comm_status_value can be set to a value from 0 to 127, where each bit corresponds to a communications error received from.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_COMM_STATUS(comm_status)

The lexical elements of the Builtin XMTR_ABORT_ON_COMM_STATUS are shown in Table B.211.

Table B.211 – Builtin XMTR_ABORT_ON_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
comm_status	int	I	m	specifies the new comm status abort mask
<return>	VOID	—	—	—

B.212 Builtin XMTR_ABORT_ON_DATA

Purpose

The Builtin XMTR_ABORT_ON_DATA will set the correct bit(s) in the abort data mask such that the method will be aborted if the specified bit is set in the data field returned by command 48.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_DATA(data)

The lexical elements of the Builtin XMTR_ABORT_ON_DATA are shown in Table B.212.

Table B.212 – Builtin XMTR_ABORT_ON_DATA

Parameter name	Type	Direction	Usage	Description
data	int	I	m	specifies the new data abort mask
<return>	VOID	—	—	—

B.213 Builtin XMTR_ABORT_ON_DEVICE_STATUS

Purpose

The Builtin XMTR_ABORT_ON_DEVICE_STATUS will set the correct bit(s) in the command 48 device status abort mask in such a way that the specified command 48 device status value will cause the method to abort. Device status is defined to be the second data octet returned in a transaction. The device_status value can be set to a value from 0 to 127, where each bit corresponds to a device status returned from the device.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

Lexical structure

XMTR_ABORT_ON_DEVICE_STATUS(device_status)

The lexical elements of the Builtin XMTR_ABORT_ON_DEVICE_STATUS are shown in Table B.213.

Table B.213 – Builtin XMTR_ABORT_ON_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
device_status	int	I	m	specifies the new device status abort mask
<return>	VOID	—	—	—

B.214 Builtin XMTR_ABORT_ON_NO_DEVICE

Purpose

The Builtin XMTR_ABORT_ON_NO_DEVICE will set the command 48 no device mask such that the method will be aborted if no device is found while sending a command 48.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_NO_DEVICE (VOID)

The lexical element of the Builtin XMTR_ABORT_ON_NO_DEVICE is shown in Table B.214.

Table B.214 – Builtin XMTR_ABORT_ON_NO_DEVICE

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.215 Builtin XMTR_ABORT_ON_RESPONSE_CODE

Purpose

The Builtin XMTR_ABORT_ON_RESPONSE_CODE will set the correct bit(s) in the response code abort mask such that the specified command 48 response code value will cause the method to abort. The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_ABORT_ON_RESPONSE_CODE(response_code)

The lexical elements of the Builtin XMTR_ABORT_ON_RESPONSE_CODE are shown in Table B.215.

Table B.215 – Builtin XMTR_ABORT_ON_RESPONSE_CODE

Parameter name	Type	Direction	Usage	Description
response_code	int	l	m	specifies the new response code abort mask
<return>	VOID	—	—	—

B.216 Builtin XMTR_IGNORE_ALL_COMM_STATUS

Purpose

The Builtin XMTR_IGNORE_ALL_COMM_STATUS will clear all of the bits in the command 48 comm status retry and abort masks. This will cause the system to ignore all bits in the comm status value. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is set.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins `send_command`, `send_command_trans`, `ext_send_command`, `ext_send_command_trans`, `get_more_status`, and `display`.

Lexical structure

`XMTR_IGNORE_ALL_COMM_STATUS(VOID)`

The lexical element of the Builtin `XMTR_IGNORE_ALL_COMM_STATUS` is shown in Table B.216.

Table B.216 – Builtin `XMTR_IGNORE_ALL_COMM_STATUS`

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.217 Builtin `XMTR_IGNORE_ALL_DEVICE_STATUS`

Purpose

The Builtin `XMTR_IGNORE_ALL_DEVICE_STATUS` will clear all of the bits in the command 48 device status retry and abort masks. This will cause the system to ignore all bits in the command 48 device status octet. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the `send_command` function for implementation of the masks. See Builtin `ABORT_ON_RESPONSE_CODE` for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins `send_command`, `send_command_trans`, `ext_send_command`, `ext_send_command_trans`, `get_more_status`, and `display`.

Lexical structure

`XMTR_IGNORE_ALL_DEVICE_STATUS(VOID)`

The lexical element of the Builtin `XMTR_IGNORE_ALL_DEVICE_STATUS` is shown in Table B.217.

Table B.217 – Builtin `XMTR_IGNORE_ALL_DEVICE_STATUS`

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.218 Builtin `XMTR_IGNORE_ALL_RESPONSE_CODES`

Purpose

The Builtin `XMTR_IGNORE_ALL_RESPONSE_CODES` will clear all of the bits in the command 48 response code retry and abort masks. This will cause the system to ignore all response code values returned from the device.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the `send_command` function for implementation of the masks. See Builtin `ABORT_ON_RESPONSE_CODE` for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_IGNORE_ALL_RESPONSE_CODES (VOID)

The lexical element of the Builtin XMTR_IGNORE_ALL_RESPONSE_CODES is shown in Table B.218.

Table B.218 – Builtin XMTR_IGNORE_ALL_RESPONSE_CODES

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.219 Builtin XMTR_IGNORE_COMM_ERROR

Purpose

The Builtin XMTR_IGNORE_COMM_ERROR will set the command 48 comm error mask such that a communications error condition will be ignored while sending command 48.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See ABORT_ON_RESPONSE_CODE for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_IGNORE_COMM_ERROR (VOID)

The lexical element of the Builtin XMTR_IGNORE_COMM_ERROR is shown in Table B.219.

Table B.219 – Builtin XMTR_IGNORE_COMM_ERROR

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.220 Builtin XMTR_IGNORE_COMM_STATUS

Purpose

The Builtin XMTR_IGNORE_COMM_STATUS will clear the correct bit(s) in the command 48 comm status abort and retry mask such that the specified bits in the command 48 comm status value will be ignored. Comm status is defined to be first data octet returned in a transaction, when bit 7 of this octet is 1.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins `send_command`, `send_command_trans`, `ext_send_command`, `ext_send_command_trans`, `get_more_status`, and `display`.

Lexical structure

`XMTR_IGNORE_COMM_STATUS(comm_status)`

The lexical elements of the Builtin `XMTR_IGNORE_COMM_STATUS` are shown in Table B.220.

Table B.220 – Builtin XMTR_IGNORE_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
<code>comm_status</code>	int	l	m	specifies the new comm status abort and retry mask
<return>	VOID	—	—	—

B.221 Builtin XMTR_IGNORE_DEVICE_STATUS

Purpose

The Builtin `XMTR_IGNORE_DEVICE_STATUS` will clear the correct bit(s) in the command 48 device status abort and retry masks such that the specified bits in the command 48 device status octet are ignored. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the `send_command` function for implementation of the masks. See `ABORT_ON_RESPONSE_CODES` for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins `send_command`, `send_command_trans`, `ext_send_command`, `ext_send_command_trans`, `get_more_status`, and `display`.

Lexical structure

`XMTR_IGNORE_DEVICE_STATUS(device_status)`

The lexical elements of the Builtin `XMTR_IGNORE_DEVICE_STATUS` are shown in Table B.221.

Table B.221 – Builtin XMTR_IGNORE_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
<code>device_status</code>	int	l	m	specifies the new device status abort and retry mask
<return>	VOID	—	—	—

B.222 Builtin XMTR_IGNORE_NO_DEVICE

Purpose

The Builtin `XMTR_IGNORE_NO_DEVICE` will set the command 48 no device mask to show that the no device condition should be ignored while sending a command 48.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the `send_command` function for implementation of the masks. See `ABORT_ON_RESPONSE_CODES` for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins `send_command`, `send_command_trans`, `ext_send_command`, `ext_send_command_trans`, `get_more_status`, and `display`.

Lexical structure

`XMTR_IGNORE_NO_DEVICE (VOID)`

The lexical element of the Builtin `XMTR_IGNORE_NO_DEVICE` is shown in Table B.222.

Table B.222 – Builtin `XMTR_IGNORE_NO_DEVICE`

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.223 Builtin `XMTR_IGNORE_RESPONSE_CODE`

Purpose

The Builtin `XMTR_IGNORE_RESPONSE_CODE` will clear the correct bit(s) in the command 48 response code masks in such a way that the specified command 48 response code value will be ignored. The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the `send_command` function for implementation of the masks. See `ABORT_ON_RESPONSE_CODES` for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins `send_command`, `send_command_trans`, `ext_send_command`, `ext_send_command_trans`, `get_more_status`, and `display`.

Lexical structure

`XMTR_IGNORE_RESPONSE_CODE (response_code)`

The lexical elements of the Builtin `XMTR_IGNORE_RESPONSE_CODE` are shown in Table B.223.

Table B.223 – Builtin `XMTR_IGNORE_RESPONSE_CODE`

Parameter name	Type	Direction	Usage	Description
<code>response_code</code>	int	I	m	specifies the new response code mask
<return>	VOID	—	—	—

B.224 Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS

Purpose

The Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS will set all of the bits in the command 48 device status retry mask. This will cause the system to retry command 48 if the device returns any command 48 device status value. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_ALL_DEVICE_STATUS (VOID)

The lexical element of the Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS is shown in Table B.224.

Table B.224 – Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.225 Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE

Purpose

The Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE will set all of the bits in the command 48 comm status retry mask. This will cause the system to retry command 48 if the device returns any command 48 comm status value. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this octet is set.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_ALL_RESPONSE_CODE (VOID)

The lexical element of the Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE is shown in Table B.225.

Table B.225 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.226 Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES**Purpose**

The Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE will set all of the bits in the command 48 response code retry mask. This will cause the system command 48 to be retried if the device returns any command 48 response code value.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_ALL_RESPONSE_CODES(VOID)

The lexical element of the Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES is shown in Table B.226.

Table B.226 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.227 Builtin XMTR_RETRY_ON_COMM_ERROR**Purpose**

The Builtin XMTR_RETRY_ON_COMM_ERROR will set the command 48 no comm error mask such that command 48 will be retried if a comm error is found while sending command 48.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_COMM_ERROR(VOID)

The lexical element of the Builtin XMTR_RETRY_ON_COMM_ERROR is shown in Table B.227.

Table B.227 – Builtin XMTR_RETRY_ON_COMM_ERROR

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.228 Builtin XMTR_RETRY_ON_COMM_STATUS**Purpose**

The Builtin XMTR_RETRY_ON_COMM_STATUS will set the correct bit(s) in the command 48 comm status retry mask in such a way that the specified command 48 comm status value will cause command 48 to be retried. Comm status is defined to be the first data octet returned in a transaction, when bit 7 of this is 1.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_COMM_STATUS(comm_status)

The lexical elements of the Builtin XMTR_RETRY_ON_COMM_STATUS are shown in Table B.228.

Table B.228 – Builtin XMTR_RETRY_ON_COMM_STATUS

Parameter name	Type	Direction	Usage	Description
comm_status	int	1	m	specifies the new comm status retry mask
<return>	VOID	—	—	—

B.229 Builtin XMTR_RETRY_ON_DATA**Purpose**

The Builtin XMTR_RETRY_ON_DATA will set the correct bit(s) in the retry data mask in such a way that the command 48 will be retried if the specified bit is set in the data field returned by command 48.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_DATA(data)

The lexical elements of the Builtin XMTR_RETRY_ON_DATA are shown in Table B.229.

Table B.229 – Builtin XMTR_RETRY_ON_DATA

Parameter name	Type	Direction	Usage	Description
data	int	l	m	specifies the new data retry mask
<return>	VOID	—	—	—

B.230 Builtin XMTR_RETRY_ON_DEVICE_STATUS

Purpose

The Builtin XMTR_RETRY_ON_DEVICE_STATUS will set the correct bit(s) in the command 48 device status retry mask in such a way that the specified command 48 device status value will cause command 48 to be retried. Device status is defined to be the second data octet returned in a transaction.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_DEVICE_STATUS(device_status)

The lexical elements of the Builtin XMTR_RETRY_ON_DEVICE_STATUS are shown in Table B.230.

Table B.230 – Builtin XMTR_RETRY_ON_DEVICE_STATUS

Parameter name	Type	Direction	Usage	Description
device_status	INT4	l	m	specifies the new device status retry mask
<return>	VOID	—	—	—

B.231 Builtin XMTR_RETRY_ON_NO_DEVICE

Purpose

The Builtin XMTR_RETRY_ON_NO_DEVICE will set the command 48 no-device mask in such a way that command 48 will be retried if no device is found while sending command 48.

The retry and abort masks are reset to their values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_NO_DEVICE (VOID)

The lexical element of the Builtin XMTR_RETRY_ON_NO_DEVICE is shown in Table B.231.

Table B.231 – Builtin XMTR_RETRY_ON_NO_DEVICE

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.232 Builtin XMTR_RETRY_ON_RESPONSE_CODE

Purpose

The Builtin XMTR_RETRY_ON_RESPONSE_CODE will set the correct bit(s) in the response code retry mask in such a way that the specified command 48 response code value will cause command 48 to be retried. The response code is defined to be the first data octet returned in a transaction, when bit 7 of this octet is 0.

The retry and abort masks are reset to their default values at the start of each method, so the new mask value will only be valid during the current method. See the send_command function for implementation of the masks. See Builtin ABORT_ON_RESPONSE_CODES for a list of the available masks and their default values.

The mask affected by this Builtin is used by the Builtins send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status, and display.

Lexical structure

XMTR_RETRY_ON_RESPONSE_CODE(response_code)

The lexical element of the Builtin XMTR_RETRY_ON_RESPONSE_CODE is shown in Table B.232.

Table B.232 – Builtin XMTR_RETRY_ON_RESPONSE_CODE

Parameter name	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.233 Builtin YearMonthDay_to_Date

Purpose

The Builtin Date will return the number of days since 1 January 1900. This value can be assigned to a date variable or can be compared to it.

Lexical structure

YearMonthDay_to_Date(year, month, day_of_month)

The lexical element of the Builtin YearMonthDay_to_Date is shown in Table B.233.

Table B.233 – Builtin YearMonthDay_to_Date

Parameter name	Type	Direction	Usage	Description
year	int	I	m	specifies the year
month	int	I	m	specifies the month
day_of_month	int	I	m	specifies the day of the month
<return>	long	O	m	returns the number of days since 1 January 1900. The EDD application shall support an implicit typecast between long and DATE

B.234 Builtins return codes

Some Builtins have specific return codes. These return codes are specified in Table B.235. The description of the Table B.235 is in Table B.234

Table B.234 – Contents of the return codes description table

Column	Meaning
Value	specifies a long integer returned from the Builtin
Identifier	specifies a reference to a returned long integer value
Description	specifies a description for the return code

Table B.235 – Return code description

Value	Identifier	Description
0	BLTIN_SUCCESS	Success
1801	BLTIN_NO_MEMORY	Out of memory
1802	BLTIN_VAR_NOT_FOUND	Cannot find variable
1803	BLTIN_BAD_ID	Specified item or member ID does not exist
1804	BLTIN_NO_DATA_TO_SEND	No value to send to device
1805	BLTIN_WRONG_DATA_TYPE	Wrong variable type for Builtin
1806	BLTIN_NO_RESP_CODES	Item does not have response codes
1807	BLTIN_BAD_METHOD_ID	Invalid method ID
1808	BLTIN_BUFFER_TOO_SMALL	Supplied buffer for starting string is too small
1809	BLTIN_CANNOT_READ_VARIABLE	Cannot read the specified variable
1810	BLTIN_INVALID_PROMPT	Bad prompt string
1811	BLTIN_NO_LANGUAGE_STRING	No valid string for current language
1812	BLTIN_DDS_ERROR	A DDS error occurred
1813	BLTIN_FAIL_RESPONSE_CODE	Response code received from the device caused Builtin to fail
1814	BLTIN_FAIL_COMM_ERROR	Communication error caused Builtin to fail
1815	BLTIN_NOT_IMPLEMENTED	Builtin function not yet implemented
1816	BLTIN_BAD_ITEM_TYPE	Wrong item type for Builtin
1817	BLTIN_VALUE_NOT_SET	A CLASS LOCAL variable has not been initialized before being used

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 9.2.
- There are the general sections for one Temperature Technology Block, one Analog Input Function Block and one Device Block. Each section starts with the block description (block_b type) which is followed by its parameters.
- Each parameter is described by its variable with attributes and its command describing the communication transactions.
- The menu section of the description describes the structure of the operator interface. The design of the screen is tool specific.

Parameter	Value	Unit	State	Name in DD
Main_MENU				Table_Main_Specialist
>> Menu_Operator				Tab_s_Operator
DEVICE_ID	IEC61804 TEMP TR		initial	DeviceBlock__DEVICE_ID
OUT_Value	0.000		valid	AnalogInputFB__OUT__Value
OUT_Status	good		valid	AnalogInputFB__OUT__Status
OUT_SCALE_UnitsIndex	degree_Celsius		initial	AnalogInputFB__OUT_SCALE__UnitsIndex
>> Menu_Specialist				Tab_s_Specialist
>> Menu_AnalogInputFunctionBlock				Tab_s_AnalogInputFunctionBlock
OUT_Value	0.000		valid	AnalogInputFB__OUT__Value
OUT_Status	good		valid	AnalogInputFB__OUT__Status
OUT_SCALE_EUat100percent	100.000		initial	AnalogInputFB__OUT_SCALE__EUat100percent
OUT_SCALE_EUat0percent	0.000		initial	AnalogInputFB__OUT_SCALE__EUat0percent
OUT_SCALE_UnitsIndex	degree_Celsius		initial	AnalogInputFB__OUT_SCALE__UnitsIndex
OUT_SCALE_DecimalPoint	0		valid	AnalogInputFB__OUT_SCALE__DecimalPoint
HI_LIM	3.400000e+038		initial	AnalogInputFB__HI_LIM
LO_LIM	-3.400000e+038		initial	AnalogInputFB__LO_LIM
>> Menu_TemperatureTechnologyBlock				Tab_s_TemperatureTechnologyBlock
PRIMARY_VALUE_Value	0.000		valid	TemperatureTB__PRIMARY_VALUE__Value
PRIMARY_VALUE_Status	good		valid	TemperatureTB__PRIMARY_VALUE__Status
PRIMARY_VALUE_UNIT	degree_Celsius		initial	TemperatureTB__PRIMARY_VALUE__UNIT
SECONDARY_VALUE_1_Value	0.000		valid	TemperatureTB__SECONDARY_VALUE_1__Value
SECONDARY_VALUE_1_Status	good		valid	TemperatureTB__SECONDARY_VALUE_1__Status
SECONDARY_VALUE_2_Value	0.000		valid	TemperatureTB__SECONDARY_VALUE_2__Value
SECONDARY_VALUE_2_Status	good		valid	TemperatureTB__SECONDARY_VALUE_2__Status
LIN_TYPE	Ptxxx		valid	TemperatureTB__LIN_TYPE
SENSOR_CONNECTION	4 wires		valid	TemperatureTB__SENSOR_CONNECTION
>> Menu_DeviceBlock				Tab_s_DeviceBlock
DEVICE_MAN_ID	0		valid	DeviceBlock__DEVICE_MAN_ID
DEVICE_ID	IEC61804 TEMP TR		initial	DeviceBlock__DEVICE_ID
DEVICE_SER_Num	chdlwelskfrgwtb		initial	DeviceBlock__DEVICE_SER_Num

Figure C.1 – Example of an operator screen using EDD

IEC 1545/06

Figure C.1 shows an example how the temperature transmitter can be visualised at an operator screen. At the left side there is the menu structure (Main_MENU, Menu_Operator, Menu_Specialist). The Menu_Operator references the measurement value as well 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
    }
}
```

Annex D (normative)

Profiles of EDDL and Builtins

D.1 Profile of EDDL and Builtins

This annex contains selection tables. The selection tables define which EDDL constructs and syntax are used and which Builtins are selected by the various consortia.

The following conventions in Table D.1, Table D.2, and Table D.3 for the profiles apply.

Table D.1 – Profile selection tables

Lexical construct	Attribute	Presence	Constraints

Table D.2 – EDDL Formal Definition profile tables

Reference	Title	Presence	Constraints

Table D.3 – Contents of selection tables

Column	Text	Meaning
Lexical construct		is a lexical construct element of the EDDL
Attributes	<text>	is one of the attributes of a lexical construct element
Presence	NO	this (sub)clause is not included in the profile
	YES	this (sub)clause is fully (100 %) included in the profile in this case no further detail is given
Constraints	—	no constraints other than given in the reference document (sub)clause, or not applicable
	<text>	the text defines the constraint directly, for longer text table footnotes or table notes may be used
Reference	<#>	clause or annex number
Title	<text>	title of heading

D.2 Profiles for PROFIBUS⁴

D.2.1 EDDL profile

Table D.4 is the selection of elements from the lexical structure constructs specified in Clause 7 and used by the consortium PROFIBUS¹ International.

Table D.4 – EDDL element selection for PROFIBUS

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.1	DD_REVISION		YES	—
7.2.2.2	DEVICE_REVISION		YES	—
7.2.2.3	DEVICE_TYPE		YES	—
7.2.2.4	EDD_PROFILE		YES	—
7.2.2.5	EDD_VERSION		YES	—
7.2.2.6	MANUFACTURER		YES	—
7.2.2.7	MANUFACTURER_EXT		YES	—
7.3	AXIS		YES	—
7.3.2.1		MAX_VALUE	YES	—
7.3.2.2		MIN_VALUE	YES	—
7.3.2.3		SCALING	YES	—
7.3.2.4		CONSTANT_UNIT	YES	—
7.4.1	BLOCK_A		NO	—
7.4.2	BLOCK_B		YES	—
7.4.2.2.1		NUMBER	YES	—
7.4.2.2.2		TYPE	YES	—
7.5	CHART		YES	—
7.30.4		MEMBERS	YES	—
7.5.2.1		CYCLE_TIME	YES	—
7.5.2.2		HEIGHT	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.5.2.4		LENGTH	YES	—
7.5.2.5		TYPE	YES	—
7.5.2.3		WIDTH	YES	—
7.6	COLLECTION		YES	—
7.6.2		item_type	YES	except: BLOCK_A, VARIABLE_LIST, PROGRAM, DOMAIN
7.30.4		MEMBERS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.7	COMMAND		YES	—
7.7.2.1		OPERATION	YES	—
7.7.2.2		TRANSACTION	YES	—
7.7.2.3		INDEX	YES	—
7.7.2.4		BLOCK_B	YES	—
7.7.2.5		NUMBER	NO	—
7.7.2.6		SLOT	YES	—
7.7.2.7		CONNECTION	YES	—
7.7.2.8		HEADER	YES	—
7.7.2.9		MODULE	YES	—
7.30.5		RESPONSE_CODES	YES	—
7.8	CONNECTION		YES	—
7.8.2		APPINSTANCE	YES	—
7.9	DOMAIN		NO	—
7.10	EDIT_DISPLAY		YES	—
7.10.2.1		EDIT_ITEMS	YES	except: BLOCK_A parameters, elements of BLOCK_A

⁴ PROFIBUS logo is the registered trade mark of PROFIBUS International (PI). PI is a non-profit trade organization to support the fieldbus PROFIBUS. This information is given for the convenience of users of this standard and does not constitute an endorsement by CENELEC of the trademark holder or any of its products. Compliance to this profile does not require use of the tradename PROFIBUS. Use of the tradename PROFIBUS requires permission from the tradename holder.

Reference	Basic construct	Attribute	Presence	Constraints
				parameters
7.30.3		LABEL	YES	—
7.10.2.2		DISPLAY_ITEMS	YES	except: BLOCK_A parameters, elements of BLOCK_A parameters
7.10.2.3		POST_EDIT_ACTIONS	YES	—
7.10.2.4		PRE_EDIT_ACTIONS	YES	—
7.11	FILE		YES	—
7.30.4		MEMBERS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.12	GRAPH		YES	—
7.30.4		MEMBERS	YES	—
7.12.2.1		HEIGHT	YES	—
7.12.2.2		WIDTH	YES	—
7.12.2.3		CYCLE_TIME	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.12.2.4		X_AXIS	YES	—
7.13	GRID		YES	—
7.13.2.1		VECTORS	YES	—
7.13.2.2		HANDLING	YES	—
7.13.2.3		HEIGHT	YES	—
7.13.2.4		WIDTH	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.13.2.5		ORIENTATION	YES	—
7.13.2.6		VALIDITY	YES	—
7.14	IMAGE		YES	—
7.14.2.1		PATH	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.14.2.2		LINK	YES	—
7.14.2.3		VALIDITY	YES	—
7.15	IMPORT		YES	—
7.16	LIKE		YES	—
7.17	LIST		YES	—
7.17.2.1		TYPE	YES	—
7.17.2.2		CAPACITY	YES	—
7.17.2.3		COUNT	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.18	MENU		YES	—
7.18.2.1		ITEMS	YES	except: BLOCK_A parameters, elements of BLOCK_A parameters, qualifier DISPLAY_VALUE
7.30.3		LABEL	YES	—
7.18.2.2		ACCESS	YES	—
7.18.2.3		ENTRY	YES	—
7.18.2.3		HELP	YES	—
7.18.2.4		POST_EDIT_ACTIONS	YES	—
7.18.2.5		POST_READ_ACTIONS	YES	—
7.18.2.6		POST_WRITE_ACTIONS	YES	—
7.18.2.7		PRE_EDIT_ACTIONS	YES	—
7.18.2.8		PRE_READ_ACTIONS	YES	—
7.18.2.9		PRE_WRITE_ACTIONS	YES	—
7.18.2.10		PURPOSE	YES	—
7.18.2.11		ROLE	YES	—
7.18.2.12		STYLE	YES	—
7.18.2.13		VALIDITY	YES	—
7.19	METHOD		YES	—
7.19.2.1		ACCESS	YES	—
7.19.2.2		CLASS	YES	except: HART
7.30.1		DEFINITION	YES	—
7.30.3		LABEL	YES	—
7.30.2		HELP	YES	—
7.19.2.3		VALIDITY	YES	—
7.20	PROGRAM		NO	—
7.21	RECORD		YES	—
7.30.4		MEMBERS	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.30.3		LABEL	YES	—
7.30.2		HELP	YES	—
7.30.5		RESPONSE_CODES	YES	—
7.22	REFERENCE_ARRAY		YES	—
7.6.2		item-type	YES	except: BLOCK_A, VARIABLE_LIST, PROGRAM, DOMAIN
7.22.2		ELEMENTS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.23.2	REFRESH		YES	—
7.23.3	UNIT		YES	—
7.23.4	WRITE_AS_ONE		YES	—
7.24	RESPONSE_CODES		YES	—
7.25	SOURCE		YES	—
7.30.4		MEMBERS	YES	—
7.25.2.1		EMPHASIS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.25.2.2		LINE_COLOR	YES	—
7.25.2.3		LINE_TYPE	YES	—
7.25.2.4		Y_AXIS	YES	—
7.25.2.5		EXIT_ACTIONS	YES	—
7.25.2.6		INIT_ACTIONS	YES	—
7.25.2.7		REFRESH_ACTIONS	YES	—
7.26	VALUE_ARRAY		YES	—
7.30.2		LABEL	YES	—
7.26.2.1		NUMBER_OF_ELEMENTS	YES	—
7.26.2.2		TYPE	YES	—
7.30.2		HELP	YES	—
7.30.5		RESPONSE_CODES	YES	—
7.27	VARIABLE		YES	—
7.27.2.1		CLASS	YES	except: HART
7.30.2		LABEL	YES	—
7.27.2.2		TYPE	YES	—
7.27.2.3		CONSTANT_UNIT	YES	—
7.27.2.4		HANDLING	YES	—
7.30.2		HELP	YES	—
7.27.2.6		POST_EDIT_ACTIONS	YES	—
7.27.2.8		POST_READ_ACTIONS	YES	—
7.27.2.9		POST_WRITE_ACTIONS	YES	—
7.27.2.10		PRE_EDIT_ACTIONS	YES	—
7.27.2.11		PRE_READ_ACTIONS	YES	—
7.27.2.12		PRE_WRITE_ACTIONS	YES	—
7.27.2.14		READ_TIMEOUT	YES	—
7.27.2.13		REFRESH_ACTIONS	YES	—
7.30.5		RESPONSE_CODE	YES	—
7.27.2.15		STYLE	YES	—
7.27.2.16		VALIDITY	YES	—
7.27.2.14		WRITE_TIMEOUT	YES	—
7.28	VARIABLE_LIST		NO	—
7.29	WAVEFORM		YES	—
7.29.2.1		TYPE	YES	—
7.29.2.2		EMPHASIS	YES	—
7.29.2.3		EXIT_ACTIONS	YES	—
7.29.2.4		HANDLING	YES	—
7.30.2		HELP	YES	—
7.29.2.5		INIT_ACTIONS	YES	—
7.29.2.6		KEY_POINTS	YES	—
7.30.3		LABEL	YES	—
7.29.2.7		LINE_COLOR	YES	—
7.29.2.8		LINE_TYPE	YES	—
7.29.2.9		REFRESH_ACTIONS	YES	—
7.29.2.10		Y_AXIS	YES	—
7.31	OPEN		NO	—
7.31	CLOSE		NO	—
7.32	Conditional expressions		YES	—
7.33	Referencing		YES	—
7.34	Strings		YES	—
7.35	Expressions		YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.36	Text dictionary		YES	—
NOTE Text is only ISO Latin-1.				

D.2.2 Builtin profile

Table D.5 is the selection of Builtins specified in Annex B and used by the consortium PROFIBUS International.

The Builtins are listed in alphabetical order.

Table D.5 – Builtin profile for PROFIBUS

Reference	Builtin Name	Presence	Constraints
B.3	abort	YES	—
B.4	abort_on_all_comm_errors	NO	—
B.5	ABORT_ON_ALL_COMM_STATUS	NO	—
B.6	ABORT_ON_ALL_DEVICE_STATUS	NO	—
B.7	ABORT_ON_ALL_RESPONSE_CODES	NO	—
B.8	abort_on_all_response_codes	NO	—
B.9	abort_on_comm_error	NO	—
B.10	ABORT_ON_COMM_ERROR	NO	—
B.11	ABORT_ON_COMM_STATUS	NO	—
B.12	ABORT_ON_DEVICE_STATUS	NO	—
B.13	ABORT_ON_NO_DEVICE	NO	—
B.14	ABORT_ON_RESPONSE_CODE	NO	—
B.15	abort_on_response_code	NO	—
B.16	abs	NO	—
B.17	ACKNOWLEDGE	YES	—
B.18	acknowledge	YES	—
B.19	acos	NO	—
B.20	add_abort_method (version A)	YES	—
B.21	add_abort_method (version B)	YES	—
B.22	asin	NO	—
B.23	assign	NO	—
B.24	assign_double	YES	—
B.25	assign_float	YES	—
B.26	assign_int	YES	—
B.27	assign_var	YES	—
B.28	atan	NO	—
B.29	ATOF	NO	—
B.30	atof	YES	—
B.31	atoi	NO	—
B.32	atoi	YES	—
B.33	BUILD_MESSAGE	NO	—
B.34	cbct	NO	—
B.35	ceil	NO	—
B.36	cos	NO	—
B.37	cosh	NO	—
B.38	dassign	YES	—
B.39	Date_to_DayOfMonth	YES	—
B.40	Date_to_Month	YES	—
B.41	Date_to_Year	YES	—
B.42	DELAY	YES	—
B.43	delay	YES	—
B.44	DELAY_TIME	YES	—
B.45	delayfor	YES	—
B.46	DICT_ID	YES	—
B.47	discard_on_exit	YES	—
B.48	display	YES	—
B.49	display_builtin_error	NO	—
B.50	display_comm_error	NO	—
B.51	display_comm_status	NO	—
B.52	display_device_status	NO	—
B.53	display_dynamics	YES	—
B.54	display_message	YES	—
B.55	display_response_code	NO	—
0	display_response_status	YES	—
B.57	display_xmtr_status	NO	—

Reference	Builtin Name	Presence	Constraints
B.58	edit_device_value	YES	—
B.59	edit_local_value	YES	—
B.60	exp	NO	—
B.61	ext_send_command	YES	—
B.62	ext_send_command_trans	YES	—
B.63	fail_on_all_comm_errors	NO	—
B.64	fail_on_all_response_codes	NO	—
B.65	fail_on_comm_error	NO	—
B.66	fail_on_response_code	NO	—
B.67	fassign	YES	—
B.68	fgetval	YES	—
B.69	float_value	YES	—
B.70	floor	NO	—
B.71	fmod	NO	—
B.72	fsetval	YES	—
B.73	ftoa	YES	—
B.74	fvar_value	YES	—
B.75	get_acknowledgement	YES	—
B.76	get_comm_error	NO	—
B.77	get_comm_error_string	NO	—
B.78	get_date	YES	—
B.79	get_date_value	YES	—
B.80	get_dds_error	NO	—
B.81	GET_DEV_VAR_VALUE	YES	—
B.82	get_dev_var_value	YES	—
B.83	get_dictionary_string	YES	—
B.84	get_double	YES	—
B.85	get_double_value	YES	—
B.86	get_enum_string	NO	—
B.87	get_float	YES	—
B.88	get_float_value	YES	—
B.89	GET_LOCAL_VAR_VALUE	YES	—
B.90	get_local_var_value	YES	—
B.91	get_more_status	YES	—
B.92	get_resolve_status	NO	—
B.93	get_response_code	NO	—
B.94	get_response_code_string	NO	—
B.95	get_signed	YES	—
B.96	get_signed_value	YES	—
B.97	get_status_code_string	NO	—
B.98	get_status_string	NO	—
B.99	get_stddict_string	YES	—
B.100	get_string	YES	—
B.101	get_string_value	YES	—
B.102	GET_TICK_COUNT	NO	—
B.103	get_unsigned	YES	—
B.104	get_unsigned_value	YES	—
B.105	lassign	YES	—
B.106	lgetval	YES	—
B.107	IGNORE_ALL_COMM_STATUS	NO	—
B.108	IGNORE_ALL_DEVICE_STATUS	NO	—
B.109	IGNORE_ALL_RESPONSE_CODES	NO	—
B.110	IGNORE_COMM_ERROR	NO	—
B.111	IGNORE_COMM_STATUS	NO	—
B.112	IGNORE_DEVICE_STATUS	NO	—
B.113	IGNORE_NO_DEVICE	NO	—
B.114	IGNORE_RESPONSE_CODE	NO	—
B.115	int_value	YES	—
B.116	is_NaN	YES	—
B.117	isetval	YES	—
B.118	ITEM_ID	NO	—
B.119	ITOA	NO	—
B.120	itoa	NO	—
B.121	ivar_value	YES	—
B.122	lassign	YES	—
B.123	lgetval	NO	—
B.124	Builtin ListDeleteElementAt	YES	—
B.125	Builtin ListInsert	YES	—
B.126	log	NO	—
B.127	log10	NO	—
B.128	log2	NO	—
B.129	LOG_MESSAGE	NO	—

Reference	Builtin Name	Presence	Constraints
B.130	long_value	YES	—
B.131	lsetval	YES	—
B.132	lvar_value	YES	—
B.133	MEMBER_ID	NO	—
B.134	Builtin MenuDisplay	YES	—
B.135	method_abort	YES	—
B.136	ObjectReference	YES	—
B.137	pow	NO	—
B.138	process_abort	YES	—
B.139	put_date	NO	—
B.140	put_date_value	NO	—
B.141	put_double	NO	—
B.142	put_double_value	NO	—
B.143	put_float	NO	—
B.144	put_float_value	NO	—
B.145	PUT_MESSAGE	YES	—
B.146	put_message	YES	—
B.147	put_signed	NO	—
B.148	put_signed_value	NO	—
B.149	put_string	NO	—
B.150	put_string_value	NO	—
B.151	put_unsigned	NO	—
B.152	put_unsigned_value	NO	—
B.153	ReadCommand	YES	—
B.154	read_value	YES	—
B.155	remove_abort_method (version A)	YES	—
B.156	remove_abort_method (version B)	YES	—
B.157	remove_all_abort_methods	YES	—
B.158	resolve_array_ref	NO	—
B.159	resolve_block_ref	NO	—
B.160	resolve_param_list_ref	NO	—
B.161	resolve_param_ref	NO	—
B.162	resolve_record_ref	NO	—
B.163	retry_on_all_comm_errors	NO	—
B.164	RETRY_ON_ALL_COMM_STATUS	NO	—
B.165	RETRY_ON_ALL_DEVICE_STATUS	NO	—
B.166	RETRY_ON_ALL_RESPONSE_CODES	NO	—
B.167	retry_on_all_response_codes	NO	—
B.168	RETRY_ON_COMM_ERROR	NO	—
B.169	retry_on_comm_error	NO	—
B.170	RETRY_ON_COMM_STATUS	NO	—
B.171	RETRY_ON_DEVICE_STATUS	NO	—
B.172	RETRY_ON_NO_DEVICE	NO	—
B.173	RETRY_ON_RESPONSE_CODE	NO	—
B.174	retry_on_response_code	NO	—
B.175	rspcode_string	NO	—
B.177	save_on_exit	YES	—
B.178	save_values	YES	—
B.179	SELECT_FROM_LIST	YES	—
B.180	select_from_list	YES	—
B.181	select_from_menu	YES	—
B.182	send	YES	—
B.183	send_all_values	YES	—
B.184	send_command	YES	—
B.185	send_command_trans	YES	—
B.186	send_on_exit	YES	—
B.187	send_trans	NO	—
B.188	send_value	YES	—
B.189	SET_NUMBER_OF_RETRIES	YES	—
B.190	sin	NO	—
B.191	sinh	NO	—
B.192	sqrt	NO	—
B.193	strcmp	NO	—
B.194	strlen	NO	—
B.195	strlwr	NO	—
B.196	strmid	NO	—
B.197	strstr	NO	—
B.198	strtrim	NO	—
B.199	strupr	NO	—
B.200	tan	NO	—
B.201	tanh	NO	—
B.202	To_Date_and_Time	NO	—

Reference	Builtin Name	Presence	Constraints
B.203	trunc	NO	—
B.204	VARID	YES	—
B.205	vassign	YES	—
B.206	WRITE_COMMAND	NO	—
B.207	XMTR_ABORT_ON_ALL_COMM_STATUS	NO	—
B.208	XMTR_ABORT_ON_ALL_DEVICE_STATUS	NO	—
B.209	XMTR_ABORT_ON_ALL_RESPONSE_CODES	NO	—
B.210	XMTR_ABORT_ON_COMM_ERROR	NO	—
B.211	XMTR_ABORT_ON_COMM_STATUS	NO	—
B.212	XMTR_ABORT_ON_DATA	NO	—
B.213	XMTR_ABORT_ON_DEVICE_STATUS	NO	—
B.214	XMTR_ABORT_ON_NO_DEVICE	NO	—
B.215	XMTR_ABORT_ON_RESPONSE_CODE	NO	—
B.216	XMTR_IGNORE_ALL_COMM_STATUS	NO	—
B.217	XMTR_IGNORE_ALL_DEVICE_STATUS	NO	—
B.218	XMTR_IGNORE_ALL_RESPONSE_CODES	NO	—
B.219	XMTR_IGNORE_COMM_ERROR	NO	—
B.220	XMTR_IGNORE_COMM_STATUS	NO	—
B.221	XMTR_IGNORE_DEVICE_STATUS	NO	—
B.222	XMTR_IGNORE_NO_DEVICE	NO	—
B.223	XMTR_IGNORE_RESPONSE_CODE	NO	—
B.224	XMTR_RETRY_ON_ALL_DEVICE_STATUS	NO	—
B.225	XMTR_RETRY_ON_ALL_RESPONSE_CODE	NO	—
B.226	XMTR_RETRY_ON_ALL_RESPONSE_CODES	NO	—
B.227	XMTR_RETRY_ON_COMM_ERROR	NO	—
B.228	XMTR_RETRY_ON_COMM_STATUS	NO	—
B.229	XMTR_RETRY_ON_DATA	NO	—
B.230	XMTR_RETRY_ON_DEVICE_STATUS	NO	—
B.231	XMTR_RETRY_ON_NO_DEVICE	NO	—
B.232	XMTR_RETRY_ON_RESPONSE_CODE	NO	—
B.233	YearMonthDay to Date	YES	—
B.234	Return Codes	YES	—

D.2.3 EDDL Formal Definition profile

Reference	Title	Presence	Constraints
A.1	EDDL Preprocessor	YES	—
A.2	Conventions	YES	—
A.3	Operators	YES	—
A.4	Keywords	YES	—
A.5	Terminals	YES	—
A.6	Formal EDDL syntax	YES	—

D.3 Profiles for FOUNDATION®⁵ Fieldbus

D.3.1 EDDL profile

Table D.6 is the selection of elements from the lexical structure constructs specified in Clause 7 and used by the consortium Fieldbus Foundation.

Table D.6 – EDDL element selection for Fieldbus Foundation

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.1		DD_REVISION	YES	—
7.2.2.2		DEVICE_REVISION	YES	—
7.2.2.3		DEVICE_TYPE	YES	—
7.2.2.4		EDD_PROFILE	NO	—
7.2.2.5		EDD_VERSION	NO	—

⁵ FOUNDATION® Fieldbus is the trade name of the consortium Fieldbus Foundation (non-profit organization). This information is given for the convenience of users of this standard and does not constitute an endorsement by CENELEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.6		MANUFACTURER	YES	—
7.2.2.7		MANUFACTURER_EXT	NO	—
7.3	AXIS		YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.3.2.1		MAX_VALUE	YES	—
7.3.2.2		MIN_VALUE	YES	—
7.3.2.3		SCALING	YES	—
7.3.2.4		CONSTANT_UNIT	YES	—
7.4.1	BLOCK_A		YES	—
7.4.1.2.1		CHARACTERISTICS	YES	—
7.30.3		LABEL	YES	—
7.4.1.2.2		PARAMETERS	YES	—
7.4.1.2.3		AXIS_ITEMS	YES	—
7.4.1.2.4		CHART_ITEMS	YES	—
7.4.1.2.5		COLLECTION_ITEMS	YES	—
7.4.1.2.6		EDIT_DISPLAY_ITEMS	YES	—
7.4.1.2.7		FILE_ITEMS	YES	—
7.4.1.2.8		GRAPH_ITEMS	YES	—
7.4.1.2.9		GRID_ITEMS	YES	—
7.4.1.2.10		IMAGE_ITEMS	YES	—
7.4.1.2.11		LIST_ITEMS	YES	—
7.4.1.2.12		LOCAL_PARAMETERS	YES	—
7.30.2		HELP	YES	—
7.4.1.2.13		MENU_ITEMS	YES	—
7.4.1.2.14		METHOD_ITEMS	YES	—
7.4.1.2.15		PARAMETER_LISTS	YES	—
7.4.1.2.16		REFERENCE_ARRAY_ITEMS	YES	Referred as ITEM_ARRAY_ITEMS
7.4.1.2.17		REFRESH_ITEMS	YES	—
7.4.1.2.18		SOURCE_ITEMS	YES	—
7.4.1.2.19		UNIT_ITEMS	YES	—
7.4.1.2.20		WAVEFORM_ITEMS	YES	—
7.4.1.2.21		WRITE_AS_ONE_ITEMS	YES	—
7.4.2	BLOCK_B		NO	—
7.5	CHART		YES	—
7.30.4		MEMBERS	YES	—
7.5.2.1		CYCLE_TIME	YES	—
7.5.2.2		HEIGHT	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.5.2.4		LENGTH	YES	—
7.5.2.5		TYPE	YES	—
7.5.2.3		WIDTH	YES	—
7.6	COLLECTION		YES	—
7.6.2		item-type	YES	If optional item-type specified, only: COLLECTION, DOMAIN, EDIT_DISPLAY, MENU, METHOD, PROGRAM, RECORD, REFERENCE_ARRAY, REFRESH, RESPONSE_CODES, UNIT, VALUE_ARRAY, VARIABLE, VARIABLE_LIST, WRITE_AS_ONE
7.30.4		MEMBERS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.7	COMMAND		NO	—
7.30.5		RESPONSE_CODES	NO	only: referenced RESPONSE_CODES
7.8	CONNECTION		NO	—
7.9	DOMAIN		YES	—
7.9.2		HANDLING	YES	—
7.30.5		RESPONSE_CODES	YES	only: referenced RESPONSE_CODES
7.10	EDIT_DISPLAY		YES	—
7.10.2.1		EDIT_ITEMS	YES	—
7.30.3		LABEL	YES	—
7.10.2.2		DISPLAY_ITEMS	YES	—
7.10.2.3		POST_EDIT_ACTIONS	YES	except: DEFINITION
7.10.2.4		PRE_EDIT_ACTIONS	YES	except: DEFINITION

Reference	Basic construct	Attribute	Presence	Constraints
7.11	FILE		YES	—
7.30.4		MEMBERS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.12	GRAPH		YES	—
7.30.4		MEMBERS	YES	—
7.12.2.1		HEIGHT	YES	—
7.12.2.2		WIDTH	YES	—
7.12.2.3		CYCLE_TIME	NO	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.12.2.4		X_AXIS	YES	—
7.13	GRID		YES	—
7.13.2.1		VECTORS	YES	—
7.13.2.2		HANDLING	YES	—
7.13.2.3		HEIGHT	YES	—
7.13.2.4		WIDTH	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.13.2.5		ORIENTATION	YES	—
7.13.2.6		VALIDITY	YES	—
7.14	IMAGE		YES	—
7.14.2.1		PATH	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.14.2.2		LINK	YES	—
7.14.2.3		VALIDITY	YES	—
7.15	IMPORT		YES	—
7.16	LIKE		YES	the optional item-type shall be used
7.17	LIST		YES	—
7.17.2.1		TYPE	YES	—
7.17.2.2		CAPACITY	YES	—
7.17.2.3		COUNT	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.18	MENU		YES	—
7.18.2.1		ITEMS	YES	—
7.30.3		LABEL	YES	—
7.18.2.2		ACCESS	NO	—
7.18.2.3		ENTRY	NO	—
7.18.2.3		HELP	NO	—
7.18.2.4		POST_EDIT_ACTIONS	NO	except: DEFINITION
7.18.2.5		POST_READ_ACTIONS	NO	except: DEFINITION
7.18.2.6		POST_WRITE_ACTIONS	NO	except: DEFINITION
7.18.2.7		PRE_EDIT_ACTIONS	NO	except: DEFINITION
7.18.2.8		PRE_READ_ACTIONS	NO	except: DEFINITION
7.18.2.9		PRE_WRITE_ACTIONS	NO	except: DEFINITION
7.18.2.10		PURPOSE	NO	—
7.18.2.11		ROLE	NO	—
7.18.2.12		STYLE	YES	except: string
7.18.2.13		VALIDITY	NO	—
7.19	METHOD		YES	—
7.19.2.1		ACCESS	NO	—
7.19.2.2		CLASS	YES	—
7.30.1		DEFINITION	YES	—
7.30.3		LABEL	YES	—
7.30.2		HELP	YES	—
7.19.2.3		TYPE	NO	—
7.19.2.4		VALIDITY	YES	—
7.20	PROGRAM		YES	—
7.20.2.1		ARGUMENT	YES	—
7.30.5		RESPONSE_CODES	YES	only: referenced RESPONSE_CODES
7.21	RECORD		YES	—
7.30.4		MEMBERS	YES	—
7.30.3		LABEL	YES	—
7.30.2		HELP	YES	—
7.30.5		RESPONSE_CODES	YES	only: referenced RESPONSE_CODES

Reference	Basic construct	Attribute	Presence	Constraints
7.22	REFERENCE_ARRAY		YES	Referred as ITEM_ARRAY
7.6.2		item-type	YES	only: COLLECTION, DOMAIN, EDIT_DISPLAY, MENU, METHOD, PROGRAM, RECORD, REFERENCE_ARRAY, REFRESH, RESPONSE_CODES, UNIT, VALUE_ARRAY, VARIABLE, VARIABLE_LIST, WRITE_AS_ONE
7.22.2		ELEMENTS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.23.2	REFRESH		YES	—
7.23.3	UNIT		YES	—
7.23.4	WRITE_AS_ONE		YES	—
7.24	RESPONSE_CODES		YES	—
7.25	SOURCE		YES	—
7.30.4		MEMBERS	YES	—
7.25.2.1		EMPHASIS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.25.2.2		LINE_COLOR	YES	—
7.25.2.3		LINE_TYPE	YES	—
7.25.2.4		Y_AXIS	YES	—
7.25.2.5		EXIT_ACTIONS	YES	—
7.25.2.6		INIT_ACTIONS	YES	—
7.25.2.7		REFRESH_ACTIONS	YES	—
7.26	VALUE_ARRAY		YES	—
7.30.2		LABEL	YES	—
7.26.2.1		NUMBER_OF_ELEMENTS	YES	—
7.26.2.2		TYPE	YES	—
7.30.2		HELP	YES	—
7.30.5		RESPONSE_CODES	YES	only: referenced RESPONSE_CODES
7.27	VARIABLE		YES	—
7.27.2.1		CLASS	YES	Every VARIABLE shall have one and only one of the following classes: CONTAINED, INPUT, OUTPUT Additional classes which are optional: ALARM, DIAGNOSTIC, DYNAMIC, LOCAL, OPERATE, SERVICE, TUNE
7.27.2.4		DEFAULT_VALUE	NO	—
7.27.2.6		INITIAL_VALUE	NO	—
7.30.3		LABEL	YES	—
7.27.2.2		TYPE	YES	except: DATE, OBJECT REFERENCE, PACKED_ASCII
7.27.2.3		CONSTANT_UNIT	YES	—
7.27.2.5		HANDLING	YES	—
7.30.2		HELP	YES	—
7.27.2.7		POST_EDIT_ACTIONS	YES	except: DEFINITION
7.27.2.8		POST_READ_ACTIONS	YES	except: DEFINITION
7.27.2.9		POST_WRITE_ACTIONS	YES	except: DEFINITION
7.27.2.10		PRE_EDIT_ACTIONS	YES	except: DEFINITION
7.27.2.11		PRE_READ_ACTIONS	YES	except: DEFINITION
7.27.2.12		PRE_WRITE_ACTIONS	YES	except: DEFINITION
7.27.2.14		READ_TIMEOUT	YES	—
7.27.2.13		REFRESH_ACTIONS	YES	—
7.30.5		RESPONSE_CODE	YES	only: referenced RESPONSE_CODES
7.27.2.15		STYLE	NO	—
7.27.2.16		VALIDITY	YES	—
7.27.2.14		WRITE_TIMEOUT	YES	—
7.28	VARIABLE_LIST		YES	—
7.30.4		MEMBERS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.30.5		RESPONSE_CODES	YES	only: referenced RESPONSE_CODES
7.29	WAVEFORM		YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.29.2.1		TYPE	YES	—
7.29.2.2		EMPHASIS	YES	—
7.29.2.3		EXIT_ACTIONS	YES	—
7.29.2.4		HANDLING	YES	—
7.30.2		HELP	YES	—
7.29.2.5		INIT_ACTIONS	YES	—
7.29.2.6		KEY_POINTS	YES	—
7.30.3		LABEL	YES	—
7.29.2.7		LINE_COLOR	YES	—
7.29.2.8		LINE_TYPE	YES	—
7.29.2.9		REFRESH_ACTIONS	YES	—
7.29.2.10		Y_AXIS	YES	—
7.31	OPEN		YES	—
7.31	CLOSE		YES	—
7.32	Conditional expressions		YES	—
7.33	Reference		YES	except: referencing HELP and LABEL attributes (see 7.34.5) of EDD instances
7.34	Strings		YES	string operations (see 7.34.6) are not possible
7.34.7	Expressions		YES	except: ARRAY_INDEX (see Table 204) and attribute values of VARIABLES: SCALING_FACTOR, DEFAULT_VALUE, INITIAL_VALUE, VARIABLE_STATUS (see 205) are not possible
7.36	Text dictionary		YES	—
NOTE Text is only ISO Latin-1.				

D.3.2 Builtin profile

Table D.7 is the selection of Builtins specified in Annex B and used by the consortium Fieldbus Foundation.

The Builtins are listed in alphabetical order.

Table D.7 – Builtin profile for Fieldbus Foundation

Reference	Builtin Name	Presence	Constraints
B.3	abort	NO	—
B.4	abort_on_all_comm_errors	YES	—
B.5	ABORT_ON_ALL_COMM_STATUS	NO	—
B.6	ABORT_ON_ALL_DEVICE_STATUS	NO	—
B.7	ABORT_ON_ALL_RESPONSE_CODES	NO	—
B.8	abort_on_all_response_codes	YES	—
B.9	abort_on_comm_error	YES	—
B.10	ABORT_ON_COMM_ERROR	NO	—
B.11	ABORT_ON_COMM_STATUS	NO	—
B.12	ABORT_ON_DEVICE_STATUS	NO	—
B.13	ABORT_ON_NO_DEVICE	NO	—
B.14	ABORT_ON_RESPONSE_CODE	NO	—
B.15	abort_on_response_code	YES	—
B.16	abs	NO	—
B.17	ACKNOWLEDGE	NO	—
B.18	acknowledge	NO	—
B.19	acos	NO	—
B.20	add_abort_method (version A)	NO	—
B.21	add_abort_method (version B)	YES	—
B.22	asin	NO	—
B.23	assign	YES	—
B.24	assign_double	NO	—
B.25	assign_float	NO	—
B.26	assign_int	NO	—
B.27	assign_var	NO	—
B.28	atan	NO	—

Reference	Builtin Name	Presence	Constraints
B.29	ATOF	NO	—
B.30	atof	NO	—
B.31	ATOI	NO	—
B.32	atoi	NO	—
B.33	BUILD_MESSAGE	NO	—
B.34	cbirt	NO	—
B.35	ceil	NO	—
B.36	cos	NO	—
B.37	cosh	NO	—
B.38	dassign	NO	—
B.39	Date_to_DayOfMonth	NO	—
B.40	Date_to_Month	NO	—
B.41	Date_to_Year	NO	—
B.42	DELAY	NO	—
B.43	delay	NO	—
B.44	DELAY_TIME	NO	—
B.45	delayfor	YES	—
B.46	DICT_ID	YES	—
B.47	discard_on_exit	YES	—
B.48	display	NO	—
B.49	display_builtin_error	YES	—
B.50	display_comm_error	YES	—
B.51	display_comm_status	NO	—
B.52	display_device_status	NO	—
B.53	display_dynamics	YES	—
B.54	display_message	YES	—
B.55	display_response_code	YES	—
0	display_response_status	NO	—
B.57	display_xmtr_status	NO	—
B.58	edit device value	YES	—
B.59	edit local value	YES	—
B.60	exp	NO	—
B.61	ext send command	NO	—
B.62	ext send command trans	NO	—
B.63	fail_on_all_comm_errors	YES	—
B.64	fail_on_all_response_codes	YES	—
B.65	fail_on_comm_error	YES	—
B.66	fail_on_response_code	YES	—
B.67	fassign	NO	—
B.68	fgetval	NO	—
B.69	float value	NO	—
B.70	floor	NO	—
B.71	fmod	NO	—
B.72	fsetval	NO	—
B.73	rtoa	NO	—
B.74	fvar value	NO	—
B.75	get acknowledgement	YES	—
B.76	get comm error	YES	—
B.77	get comm error string	YES	—
B.78	get date	YES	—
B.79	get date value	YES	—
B.80	get dds error	YES	—
B.81	GET_DEV_VAR_VALUE	NO	—
B.82	get dev var value	NO	—
B.83	get dictionary string	NO	—
B.84	get double	YES	—
B.85	get double value	YES	—
B.86	get enum string	NO	—
B.87	get float	YES	—
B.88	get float value	YES	—
B.89	GET_LOCAL_VAR_VALUE	NO	—
B.90	get local var value	NO	—
B.91	get more status	NO	—
B.92	get resolve status	YES	—
B.93	get response code	YES	—
B.94	get response code string	YES	—
B.95	get signed	YES	—
B.96	get signed value	YES	—
B.97	get status code string	NO	—
B.98	get status string	YES	—
B.99	get stddict string	YES	—
B.100	get string	YES	—

Reference	Builtin Name	Presence	Constraints
B.101	get_string_value	YES	—
B.102	GET_TICK_COUNT	NO	—
B.103	get_unsigned	YES	—
B.104	get_unsigned_value	YES	—
B.105	iassign	NO	—
B.106	igetval	NO	—
B.107	IGNORE_ALL_COMM_STATUS	NO	—
B.108	IGNORE_ALL_DEVICE_STATUS	NO	—
B.109	IGNORE_ALL_RESPONSE_CODES	NO	—
B.110	IGNORE_COMM_ERROR	NO	—
B.111	IGNORE_COMM_STATUS	NO	—
B.112	IGNORE_DEVICE_STATUS	NO	—
B.113	IGNORE_NO_DEVICE	NO	—
B.114	IGNORE_RESPONSE_CODE	NO	—
B.115	int_value	NO	—
B.116	is_NaN	YES	—
B.117	isetval	NO	—
B.118	ITEM_ID	YES	—
B.119	ITOA	NO	—
B.120	itoa	NO	—
B.121	ivar_value	NO	—
B.122	lassign	NO	—
B.123	lgetval	NO	—
B.124	Builtin ListDeleteElementAt	YES	—
B.125	Builtin ListInsert	YES	—
B.126	log	NO	—
B.127	log10	NO	—
B.128	log2	NO	—
B.129	LOG_MESSAGE	NO	—
B.130	long_value	NO	—
B.131	lsetval	NO	—
B.132	lvar_value	NO	—
B.133	MEMBER_ID	YES	—
B.134	Builtin MenuDisplay	YES	—
B.135	method_abort	YES	—
B.136	ObjectReference	NO	—
B.137	pow	NO	—
B.138	process_abort	NO	—
B.139	put_date	YES	—
B.140	put_date_value	YES	—
B.141	put_double	YES	—
B.142	put_double_value	YES	—
B.143	put_float	YES	—
B.144	put_float_value	YES	—
B.145	PUT_MESSAGE	NO	—
B.146	put_message	NO	—
B.147	put_signed	YES	—
B.148	put_signed_value	YES	—
B.149	put_string	YES	—
B.150	put_string_value	YES	—
B.151	put_unsigned	YES	—
B.152	put_unsigned_value	YES	—
B.153	ReadCommand	NO	—
B.154	read_value	YES	—
B.155	remove_abort_method (version A)	NO	—
B.156	remove_abort_method (version B)	YES	—
B.157	remove_all_abort_methods	YES	—
B.158	resolve_array_ref	YES	—
B.159	resolve_block_ref	YES	—
B.160	resolve_param_list_ref	YES	—
B.161	resolve_param_ref	YES	—
B.162	resolve_record_ref	YES	—
B.163	retry_on_all_comm_errors	YES	—
B.164	RETRY_ON_ALL_COMM_STATUS	NO	—
B.165	RETRY_ON_ALL_DEVICE_STATUS	NO	—
B.166	RETRY_ON_ALL_RESPONSE_CODES	NO	—
B.167	retry_on_all_response_codes	YES	—
B.168	RETRY_ON_COMM_ERROR	NO	—
B.169	retry_on_comm_error	YES	—
B.170	RETRY_ON_COMM_STATUS	NO	—
B.171	RETRY_ON_DEVICE_STATUS	NO	—
B.172	RETRY_ON_NO_DEVICE	NO	—

Reference	Builtin Name	Presence	Constraints
B.173	RETRY_ON_RESPONSE_CODE	NO	—
B.174	retry_on_response_code	YES	—
B.175	round	NO	—
B.176	rspcode_string	NO	—
B.177	save_on_exit	YES	—
B.178	save_values	NO	—
B.179	SELECT_FROM_LIST	NO	—
B.180	select_from_list	NO	—
B.181	select_from_menu	YES	—
B.182	send	NO	—
B.183	send_all_values	YES	—
B.184	send_command	NO	—
B.185	send_command_trans	NO	—
B.186	send_on_exit	YES	—
B.187	send_trans	NO	—
B.188	send_value	YES	—
B.189	SET_NUMBER_OF_RETRIES	NO	—
B.190	sin	NO	—
B.191	sinh	NO	—
B.192	sqrt	NO	—
B.193	strcmp	NO	—
B.194	strlen	NO	—
B.195	strlwr	NO	—
B.196	strmid	NO	—
B.197	strstr	NO	—
B.198	strtrim	NO	—
B.199	strupr	NO	—
B.200	tan	NO	—
B.201	tanh	NO	—
B.202	To Date and Time	NO	—
B.203	trunc	NO	—
B.204	VARID	NO	—
B.205	vassign	NO	—
B.206	WRITE_COMMAND	NO	—
B.207	XMTR_ABORT_ON_ALL_COMM_STATUS	NO	—
B.208	XMTR_ABORT_ON_ALL_DEVICE_STATUS	NO	—
B.209	XMTR_ABORT_ON_ALL_RESPONSE_CODES	NO	—
B.210	XMTR_ABORT_ON_COMM_ERROR	NO	—
B.211	XMTR_ABORT_ON_COMM_STATUS	NO	—
B.212	XMTR_ABORT_ON_DATA	NO	—
B.213	XMTR_ABORT_ON_DEVICE_STATUS	NO	—
B.214	XMTR_ABORT_ON_NO_DEVICE	NO	—
B.215	XMTR_ABORT_ON_RESPONSE_CODE	NO	—
B.216	XMTR_IGNORE_ALL_COMM_STATUS	NO	—
B.217	XMTR_IGNORE_ALL_DEVICE_STATUS	NO	—
B.218	XMTR_IGNORE_ALL_RESPONSE_CODES	NO	—
B.219	XMTR_IGNORE_COMM_ERROR	NO	—
B.220	XMTR_IGNORE_COMM_STATUS	NO	—
B.221	XMTR_IGNORE_DEVICE_STATUS	NO	—
B.222	XMTR_IGNORE_NO_DEVICE	NO	—
B.223	XMTR_IGNORE_RESPONSE_CODE	NO	—
B.224	XMTR_RETRY_ON_ALL_DEVICE_STATUS	NO	—
B.225	XMTR_RETRY_ON_ALL_RESPONSE_CODE	NO	—
B.226	XMTR_RETRY_ON_ALL_RESPONSE_CODES	NO	—
B.227	XMTR_RETRY_ON_COMM_ERROR	NO	—
B.228	XMTR_RETRY_ON_COMM_STATUS	NO	—
B.229	XMTR_RETRY_ON_DATA	NO	—
B.230	XMTR_RETRY_ON_DEVICE_STATUS	NO	—
B.231	XMTR_RETRY_ON_NO_DEVICE	NO	—
B.232	XMTR_RETRY_ON_RESPONSE_CODE	NO	—
B.233	YearMonthDay_to_Date	NO	—
B.234	Return Codes	YES	—

D.3.3 EDDL Formal Definition profile

Reference	Title	Presence	Constraints
A.1	EDDL Preprocessor	YES	—
A.2	Conventions	YES	—
A.3	Operators	YES	—
A.4	Keywords	YES	—

Reference	Title	Presence	Constraints
A.5	Terminals	YES	—
A.6	Formal EDDL syntax	YES	—

D.4 Profiles for HART® Communication Foundation (HCF)⁶

D.4.1 EDDL profile

Table D.8 is the selection of elements from the lexical structure constructs specified in Clause 7 and used by the consortium HART ® Communication Foundation.

Table D.8 – EDDL element selection for HCF

Reference	Basic construct	Attribute	Presence	Constraints
7.2.2.1	DD_REVISION		YES	One octet
7.2.2.2	DEVICE_REVISION		YES	One octet
7.2.2.3	DEVICE_TYPE		YES	One octet
7.2.2.4	EDD_PROFILE		NO	—
7.2.2.5	EDD_VERSION		NO	—
7.2.2.6	MANUFACTURER		YES	Two octets
7.2.2.7	MANUFACTURER_EXT		NO	—
7.3	AXIS		YES	—
7.3.2.1		MAX_VALUE	YES	—
7.3.2.2		MIN_VALUE	YES	—
7.3.2.3		SCALING	YES	—
7.3.2.4		CONSTANT_UNIT	YES	—
7.4.1	BLOCK_A		NO	—
7.4.2	BLOCK_B		NO	—
7.5	CHART		YES	—
7.30.4		MEMBERS	YES	—
7.5.2.1		CYCLE_TIME	YES	—
7.5.2.2		HEIGHT	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.5.2.4		LENGTH	YES	—
7.5.2.5		TYPE	YES	—
7.5.2.3		WIDTH	YES	—
7.6	COLLECTION			
7.6.2		item-type	YES	only: COLLECTION, COMMANDS, EDIT_DISPLAY, MENU, METHOD, REFERENCE_ARRAY, REFRESH, RESPONSE_CODES, UNIT, VARIABLE, WRITE_AS_ONE
7.30.4		MEMBERS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.7	COMMAND		YES	—
7.7.2.1		OPERATION	YES	only: READ, WRITE, COMMAND, no conditionals
7.7.2.2		TRANSACTION	YES	No conditionals
7.7.2.3		INDEX	NO	—
7.7.2.4		BLOCK_B	NO	—
7.7.2.5		NUMBER	YES	No conditionals
7.7.2.6		SLOT	NO	—
7.7.2.7		CONNECTION	NO	—
7.7.2.8		HEADER	NO	—
7.7.2.9		MODULE	NO	—
7.30.5		RESPONSE_CODES	YES	only: embedded RESPONSE_CODES; no conditionals
7.8	CONNECTION		NO	—
7.9	DOMAIN		NO	—

⁶ HART® is a registered trademark of the consortium HART Communication Foundation (HCF) (non-profit organization). This information is given for the convenience of users of this standard and does not constitute an endorsement by CENELEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Reference	Basic construct	Attribute	Presence	Constraints
7.10	EDIT_DISPLAY		YES	—
7.10.2.1		EDIT_ITEMS	YES	only: VARIABLE, WRITE_AS_ONE
7.30.3		LABEL	YES	—
7.10.2.2		DISPLAY_ITEMS	YES	only: VARIABLE
7.10.2.3		POST_EDIT_ACTIONS	YES	except: DEFINITION
7.10.2.4		PRE_EDIT_ACTIONS	YES	except: DEFINITION
7.11	FILE		YES	—
7.30.4		MEMBERS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.12	GRAPH		YES	—
7.30.4		MEMBERS	YES	—
7.12.2.1		HEIGHT	YES	—
7.12.2.2		WIDTH	YES	—
7.12.2.3		CYCLE_TIME	NO	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.12.2.4		X_AXIS	YES	—
7.13	GRID		YES	—
7.13.2.1		VECTORS	YES	—
7.13.2.2		HANDLING	YES	—
7.13.2.3		HEIGHT	YES	—
7.13.2.4		WIDTH	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.13.2.5		ORIENTATION	YES	—
7.13.2.6		VALIDITY	YES	—
7.14	IMAGE		YES	—
7.14.2.1		PATH	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.14.2.2		LINK	YES	—
7.14.2.3		VALIDITY	YES	—
7.15	IMPORT		YES	—
7.16	LIKE		YES	—
7.17	LIST		YES	—
7.17.2.1		TYPE	YES	—
7.17.2.2		CAPACITY	YES	—
7.17.2.3		COUNT	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.18	MENU		YES	—
7.18.2.1		ITEMS	YES	except: BLOCK_A parameters, elements of BLOCK_A parameters; No HIDDEN qualifier
7.30.3		LABEL	YES	—
7.18.2.2		ACCESS	NO	—
7.18.2.3		ENTRY	NO	—
7.18.2.3		HELP	YES	—
7.18.2.4		POST_EDIT_ACTIONS	NO	—
7.18.2.5		POST_READ_ACTIONS	NO	—
7.18.2.6		POST_WRITE_ACTIONS	NO	—
7.18.2.7		PRE_EDIT_ACTIONS	NO	—
7.18.2.8		PRE_READ_ACTIONS	NO	—
7.18.2.9		PRE_WRITE_ACTIONS	NO	—
7.18.2.10		PURPOSE	NO	—
7.18.2.11		ROLE	NO	—
7.18.2.12		STYLE	YES	—
7.18.2.13		VALIDITY	YES	—
7.19	METHOD		YES	—
7.19.2.1		ACCESS	NO	—
7.19.2.2		CLASS	YES	except ALARM, CONTAINED, DYNAMIC, LOCAL, OPERATE, OUTPUT, TUNE
7.30.1		DEFINITION	YES	—
7.30.3		LABEL	YES	—
7.30.2		HELP	YES	—
7.19.2.3		VALIDITY	YES	—
7.20	PROGRAM		NO	—
7.21	RECORD		NO	—

Reference	Basic construct	Attribute	Presence	Constraints
7.22	REFERENCE_ARRAY		YES	concrete syntax: ARRAY
7.6.2		item-type	YES	—
7.22.2		ELEMENTS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.23.2	REFRESH		YES	—
7.23.3	UNIT		YES	—
7.23.4	WRITE_AS_ONE		YES	—
7.24	RESPONSE_CODES		NO	—
7.25	SOURCE		YES	—
7.30.4		MEMBERS	YES	—
7.25.2.1		EMPHASIS	YES	—
7.30.2		HELP	YES	—
7.30.3		LABEL	YES	—
7.25.2.2		LINE_COLOR	YES	—
7.25.2.3		LINE_TYPE	YES	—
7.25.2.4		Y_AXIS	YES	—
7.25.2.5		EXIT_ACTIONS	YES	—
7.25.2.6		INIT_ACTIONS	YES	—
7.25.2.7		REFRESH_ACTIONS	YES	—
7.26	VALUE_ARRAY		YES	concrete syntax: ARRAY
7.30.2		LABEL	YES	—
7.26.2.1		NUMBER_OF_ELEMENTS	YES	—
7.26.2.2		TYPE	YES	—
7.30.2		HELP	YES	—
7.30.5		RESPONSE_CODES	YES	—
7.27	VARIABLE		YES	—
7.27.2.1		CLASS	YES	Every VARIABLE shall have one and only one of the following classes: ANALOG_OUTPUT, COMPUTATION, SENSOR_CORRECTION, DEVICE, DISCRETE, FREQUENCY, HART, INPUT, LOCAL_DISPLAY Additional classes which are optional: DIAGNOSTIC, DYNAMIC, LOCAL, SERVICE
7.30.2		LABEL	YES	Is optional
7.27.2.2		TYPE	YES	only: ASCII, BITSTRING, BIT ENUMERATED, DATE_AND_TIME, DATE, DURATION, DOUBLE, ENUMERATED, FLOAT, INTEGER, INDEX, PASSWORD, PACKED_ASCII, TIME, UNSIGNED_INTEGER. No conditionals type. No INITIAL_VALUE.
7.27.2.3		CONSTANT_UNIT	YES	—
7.27.2.4		HANDLING	YES	—
7.30.2		HELP	YES	—
7.27.2.6		POST_EDIT_ACTIONS	YES	except: DEFINITION
7.27.2.8		POST_READ_ACTIONS	YES	except: DEFINITION
7.27.2.9		POST_WRITE_ACTIONS	YES	except: DEFINITION
7.27.2.10		PRE_EDIT_ACTIONS	YES	except: DEFINITION
7.27.2.11		PRE_READ_ACTIONS	YES	except: DEFINITION
7.27.2.12		PRE_WRITE_ACTIONS	YES	except: DEFINITION
7.27.2.14		READ_TIMEOUT	YES	—
7.27.2.13		REFRESH_ACTIONS	YES	—
7.30.5		RESPONSE_CODE	NO	—
7.27.2.15		STYLE	NO	—
7.27.2.16		VALIDITY	YES	—
7.27.2.14		WRITE_TIMEOUT	YES	—
7.28	VARIABLE_LIST		NO	—
7.29	WAVEFORM		YES	—
7.29.2.1		TYPE	YES	—
7.29.2.2		EMPHASIS	YES	—
7.29.2.3		EXIT_ACTIONS	YES	—
7.29.2.4		HANDLING	YES	—
7.30.2		HELP	YES	—

Reference	Basic construct	Attribute	Presence	Constraints
7.29.2.5		INIT_ACTIONS	YES	—
7.29.2.6		KEY_POINTS	YES	—
7.30.3		LABEL	YES	—
7.29.2.7		LINE_COLOR	YES	—
7.29.2.8		LINE_TYPE	YES	—
7.29.2.9		REFRESH_ACTIONS	YES	—
7.29.2.10		Y_AXIS	YES	—
7.31	OPEN		NO	—
7.31	CLOSE		NO	—
7.32	Conditional expressions		YES	
7.33	Referencing		YES	except: referencing HELP and LABEL attributes (see 7.34.5) of EDD instances
7.34	Strings		YES	string operations (see 7.34.6) are not possible
7.35	Expressions		YES	except: ARRAY_INDEX (see Table 204) and attribute values of VARIABLES: SCALING_FACTOR, INITIAL_VALUE, VARIABLE_STATUS (see 205) are not possible
7.36	Text dictionary		YES	—

NOTE Text is only ISO Latin-1 except for Katakana, (|kt|). This language is entered as Romaji and is a series of English characters. 2-3 Characters per Katakana symbol.

D.4.2 Builtin profile

Table D.9 is the selection of Builtins specified in Annex B and used by the consortium HART ® Communication Foundation (HCF).

The Builtins are listed in alphabetical order.

Table D.9 – Builtin profile for HCF

Reference	Builtin Name	Presence	Constraints
B.3	abort	YES	—
B.4	abort_on_all_comm_errors	NO	—
B.5	ABORT_ON_ALL_COMM_STATUS	YES	—
B.6	ABORT_ON_ALL_DEVICE_STATUS	YES	—
B.7	ABORT_ON_ALL_RESPONSE_CODES	YES	—
B.8	abort_on_all_response_codes	NO	—
B.9	abort_on_comm_error	NO	—
B.10	ABORT_ON_COMM_ERROR	YES	—
B.11	ABORT_ON_COMM_STATUS	YES	—
B.12	ABORT_ON_DEVICE_STATUS	YES	—
B.13	ABORT_ON_NO_DEVICE	YES	—
B.14	ABORT_ON_RESPONSE_CODE	YES	—
B.15	abort_on_response_code	NO	—
B.16	abs	YES	—
B.17	ACKNOWLEDGE	YES	—
B.18	acknowledge	YES	—
B.19	acos	YES	—
B.20	add_abort_method (version A)	YES	—
B.21	add_abort_method (version B)	NO	—
B.22	asin	YES	—
B.23	assign	NO	—
B.24	assign_double	YES	—
B.25	assign_float	YES	—
B.26	assign_int	YES	—
B.27	assign_var	YES	—
B.28	atan	YES	—
B.29	ATOF	YES	—
B.30	atof	NO	—
B.31	ATOI	YES	—
B.32	atoi	NO	—
B.33	BUILD_MESSAGE	YES	—
B.34	cbt	YES	—
B.35	ceil	YES	—

Reference	Builtin Name	Presence	Constraints
B.36	cos	YES	—
B.37	cosh	YES	—
B.38	dassign	YES	—
B.39	Date_to_DayOfMonth	NO	—
B.40	Date_to_Month	NO	—
B.41	Date_to_Year	NO	—
B.42	DELAY	YES	—
B.43	delay	YES	—
B.44	DELAY_TIME	YES	—
B.45	delayfor	NO	—
B.46	DICT_ID	NO	—
B.47	discard_on_exit	NO	—
B.48	display	YES	—
B.49	display_builtin_error	NO	—
B.50	display_comm_error	NO	—
B.51	display_comm_status	YES	—
B.52	display_device_status	YES	—
B.53	display_dynamics	NO	—
B.54	display_message	NO	—
B.55	display_response_code	NO	—
0	display_response_status	YES	—
B.57	display_xmtr_status	YES	—
B.58	edit_device_value	NO	—
B.59	edit_local_value	NO	—
B.60	exp	YES	—
B.61	ext_send_command	YES	—
B.62	ext_send_command_trans	YES	—
B.63	fail_on_all_comm_errors	NO	—
B.64	fail_on_all_response_codes	NO	—
B.65	fail_on_comm_error	NO	—
B.66	fail_on_response_code	NO	—
B.67	fassign	YES	—
B.68	fgetval	YES	—
B.69	float_value	YES	—
B.70	floor	YES	—
B.71	fmod	YES	—
B.72	fsetval	YES	—
B.73	ftoa	NO	—
B.74	fvar_value	YES	—
B.75	get_acknowledgement	NO	—
B.76	get_comm_error	NO	—
B.77	get_comm_error_string	NO	—
B.78	get_date	NO	—
B.79	get_date_value	NO	—
B.80	get_ddr_error	NO	—
B.81	GET_DEV_VAR_VALUE	YES	—
B.82	get_dev_var_value	YES	—
B.83	get_dictionary_string	YES	—
B.84	get_double	NO	—
B.85	get_double_value	NO	—
B.86	get_enum_string	YES	—
B.87	get_float	NO	—
B.88	get_float_value	NO	—
B.89	GET_LOCAL_VAR_VALUE	YES	—
B.90	get_local_var_value	YES	—
B.91	get_more_status	YES	—
B.92	get_resolve_status	NO	—
B.93	get_response_code	NO	—
B.94	get_response_code_string	NO	—
B.95	get_signed	NO	—
B.96	get_signed_value	NO	—
B.97	get_status_code_string	YES	—
B.98	get_status_string	NO	—
B.99	get_stddict_string	NO	—
B.100	get_string	NO	—
B.101	get_string_value	NO	—
B.102	GET_TICK_COUNT	NO	—
B.103	get_unsigned	NO	—
B.104	get_unsigned_value	NO	—
B.105	iassign	YES	—
B.106	igetval	YES	—

Reference	Builtin Name	Presence	Constraints
B.107	IGNORE_ALL_COMM_STATUS	YES	—
B.108	IGNORE_ALL_DEVICE_STATUS	YES	—
B.109	IGNORE_ALL_RESPONSE_CODES	YES	—
B.110	IGNORE_COMM_ERROR	YES	—
B.111	IGNORE_COMM_STATUS	YES	—
B.112	IGNORE_DEVICE_STATUS	YES	—
B.113	IGNORE_NO_DEVICE	YES	—
B.114	IGNORE_RESPONSE_CODE	YES	—
B.115	int_value	YES	—
B.116	is_NaN	NO	—
B.117	isetval	YES	—
B.118	ITEM_ID	NO	—
B.119	ITOA	YES	—
B.120	itoa	NO	—
B.121	ivar_value	YES	—
B.122	lassign	YES	—
B.123	lgetval	YES	—
B.124	Builtin ListDeleteElementAt	YES	—
B.125	Builtin ListInsert	YES	—
B.126	log	YES	—
B.127	log10	YES	—
B.128	log2	YES	—
B.129	LOG_MESSAGE	NO	—
B.130	long_value	YES	—
B.131	lsetval	YES	—
B.132	lvar_value	YES	—
B.133	MEMBER_ID	NO	—
B.134	Builtin MenuDisplay	YES	—
B.135	method_abort	NO	—
B.136	ObjectReference	NO	—
B.137	pow	YES	—
B.138	process_abort	YES	—
B.139	put_date	NO	—
B.140	put_date_value	NO	—
B.141	put_double	NO	—
B.142	put_double_value	NO	—
B.143	put_float	NO	—
B.144	put_float_value	NO	—
B.145	PUT_MESSAGE	YES	—
B.146	put_message	YES	—
B.147	put_signed	NO	—
B.148	put_signed_value	NO	—
B.149	put_string	NO	—
B.150	put_string_value	NO	—
B.151	put_unsigned	NO	—
B.152	put_unsigned_value	NO	—
B.153	ReadCommand	NO	—
B.154	read_value	NO	—
B.155	remove_abort_method (version A)	YES	—
B.156	remove_abort_method (version B)	NO	—
B.157	remove_all_abort_methods	YES	—
B.158	resolve_array_ref	NO	—
B.159	resolve_block_ref	NO	—
B.160	resolve_param_list_ref	NO	—
B.161	resolve_param_ref	NO	—
B.162	resolve_record_ref	NO	—
B.163	retry_on_all_comm_errors	NO	—
B.164	RETRY_ON_ALL_COMM_STATUS	YES	—
B.165	RETRY_ON_ALL_DEVICE_STATUS	YES	—
B.166	RETRY_ON_ALL_RESPONSE_CODES	YES	—
B.167	retry_on_all_response_codes	NO	—
B.168	RETRY_ON_COMM_ERROR	YES	—
B.169	retry_on_comm_error	NO	—
B.170	RETRY_ON_COMM_STATUS	YES	—
B.171	RETRY_ON_DEVICE_STATUS	YES	—
B.172	RETRY_ON_NO_DEVICE	YES	—
B.173	RETRY_ON_RESPONSE_CODE	YES	—
B.174	retry_on_response_code	NO	—
B.175	round	YES	—
B.176	rspcode_string	YES	—
B.177	save_on_exit	NO	—

Reference	Builtin Name	Presence	Constraints
B.178	save_values	YES	—
B.179	SELECT_FROM_LIST	YES	—
B.180	select_from_list	YES	—
B.181	select_from_menu	NO	—
B.182	send	YES	—
B.183	send_all_values	NO	—
B.184	send_command	YES	—
B.185	send_command_trans	YES	—
B.186	send_on_exit	NO	—
B.187	send_trans	YES	—
B.188	send_value	NO	—
B.189	SET_NUMBER_OF_RETRIES	YES	—
B.190	sin	YES	—
B.191	sinh	YES	—
B.192	sqrt	YES	—
B.193	strcmp	YES	—
B.194	strlen	YES	—
B.195	strlwr	YES	—
B.196	strmid	YES	—
B.197	strstr	YES	—
B.198	strtrim	YES	—
B.199	strupr	YES	—
B.200	tan	YES	—
B.201	tanh	YES	—
B.202	To Date and Time	NO	—
B.203	trunc	YES	—
B.204	VARID	YES	—
B.205	vassign	YES	—
B.206	WRITE_COMMAND	NO	—
B.207	XMTR ABORT ON ALL COMM STATUS	YES	—
B.208	XMTR ABORT ON ALL DEVICE STATUS	YES	—
B.209	XMTR ABORT ON ALL RESPONSE CODES	YES	—
B.210	XMTR ABORT ON COMM ERROR	YES	—
B.211	XMTR ABORT ON COMM STATUS	YES	—
B.212	XMTR ABORT ON DATA	YES	—
B.213	XMTR ABORT ON DEVICE STATUS	YES	—
B.214	XMTR ABORT ON NO DEVICE	YES	—
B.215	XMTR ABORT ON RESPONSE CODE	YES	—
B.216	XMTR IGNORE ALL COMM STATUS	YES	—
B.217	XMTR IGNORE ALL DEVICE STATUS	YES	—
B.218	XMTR IGNORE ALL RESPONSE CODES	YES	—
B.219	XMTR IGNORE COMM ERROR	YES	—
B.220	XMTR IGNORE COMM STATUS	YES	—
B.221	XMTR IGNORE DEVICE STATUS	YES	—
B.222	XMTR IGNORE NO DEVICE	YES	—
B.223	XMTR IGNORE RESPONSE CODE	YES	—
B.224	XMTR RETRY ON ALL DEVICE STATUS	YES	—
B.225	XMTR RETRY ON ALL RESPONSE CODE	YES	—
B.226	XMTR RETRY ON ALL RESPONSE CODES	YES	—
B.227	XMTR RETRY ON COMM ERROR	YES	—
B.228	XMTR RETRY ON COMM STATUS	YES	—
B.229	XMTR RETRY ON DATA	YES	—
B.230	XMTR RETRY ON DEVICE STATUS	YES	—
B.231	XMTR RETRY ON NO DEVICE	YES	—
B.232	XMTR RETRY ON RESPONSE CODE	YES	—
B.233	YearMonthDay_to_Date	NO	—
B.234	Return Codes	YES	—

D.4.3 EDDL Formal Definition profile

Reference	Title	Presence	Constraints
A.1	EDDL Preprocessor	YES	—
A.2	Conventions	YES	—
A.3	Operators	YES	—
A.4	Keywords	YES	—
A.5	Terminals	YES	—
A.6	Formal EDDL syntax	YES	—

D.5 Data types

D.5.1 METHOD DEFINITIONS data types

The specified size in brackets of the VARIABLE TYPES ensures that an overflow or underflow cannot occur. Table D.10 specifies the METHOD DEFINITIONS data types and Table D.11 specifies the VARIABLE TYPES data types.

Table D.10 – METHOD DEFINITIONS 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/int	-32768	+32767	—
Long	-2147483648	+2147483647	—
unsigned char	0	+255	—
unsigned short / unsigned int	0	+65535	—
unsigned long	0	+4294967295	—
float	-3.402823466e + 38	+3.402823466e + 38	—
double	-1.7976931348623157e + 308	+1.7976931348623157e + 308	—
DD_STRING	—	—	arbitrary length character string
DD_ITEM	0	+4294967295	id of any item in the EDD

Table D.11 – VARIABLE TYPES

VARIABLE TYPES	Minimum value	Maximum value
DOUBLE	-1.7976931348623157e + 308	+1.7976931348623157e + 308
FLOAT	-3.402823466e + 38	+3.402823466e + 38
INTEGER(1)	-128	+127
INTEGER(2)	-32768	+32767
INTEGER(3)	-8388608	+8388607
INTEGER(4)	-2147483648	+2147483647
UNSIGNED_INTEGER(1)	0	+255
UNSIGNED_INTEGER(2)	0	+65535
UNSIGNED_INTEGER(3)	0	+16777215
UNSIGNED_INTEGER(4)	0	+4294967295
DATE	see Table D.12	see Table D.12
DATE_AND_TIME	see Table D.13	see Table D.13
DURATION	see Table D.14	see Table D.14
TIME	see Table D.15	see Table D.15
TIME_VALUE	see Table D.16	see Table D.16
BIT_ENUMERATED(1)	0	+255

VARIABLE TYPES	Minimum value	Maximum value
BIT_ENUMERATED(2)	0	+65535
BIT_ENUMERATED(3)	0	+16777215
BIT_ENUMERATED(4)	0	+4294967295
ENUMERATED(1)	0	+255
ENUMERATED(2)	0	+65535
ENUMERATED(3)	0	+16777215
ENUMERATED(4)	0	+4294967295
INDEX(1)	0	+255
INDEX(2)	0	+65535
INDEX(3)	0	+16777215
INDEX(4)	0	+4294967295
OBJECT REFERENCE	see 7.27.2.2.4	see 7.27.2.2.4
ASCII(size)	0	+255 per character
BITSTRING(size)	0	+255 per character
EUC(size)	0	+255 per character
OCTET(size)	0	+255 per character
PACKED_ASCII	see Table D.17	see Table D.17
PASSWORD(size)	0	+255 per character
VISIBLE(size)	0	+255 per character

D.5.2 Coding of data DATE

Table D.12 specifies the DATE coding.

Table D.12 – DATE coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	1..31 day of month
2	2 ⁷	2 ⁶	2 ⁵	2 ⁴	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	1...255 years (actual year minus 1900)

D.5.3 Coding of data DATE_AND_TIME

Table D.13 specifies the DATE_AND_TIME coding.

Table D.13 – 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...59999 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	bit 6-8: 1..7 day of week (1=Monday, 7=Sunday) bit 1-5: 1..31 day of month
6	R		2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	1...12 months

Octet	Bit								Description
7	R	2^6	2^5	2^4	2^3	2^2	2^1	2^0	1...255 years (actual year minus 1900)
R means reserved for future use; S = 0 indicates standard time; S = 1 indicates summer time									

D.5.4 Coding of data DURATION

Table D.14 specifies the DURATION coding.

Table D.14 – DURATION coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of ms since midnight
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Number of days (optional)
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

D.5.5 Coding of data TIME

Table D.15 specifies the TIME coding.

Table D.15 – TIME coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds since midnight
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Number of days since 01.01.1984
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

D.5.6 Coding of data TIME_VALUE

Table D.16 specifies the TIME_VALUE coding.

Table D.16 – TIME_VALUE coding

Octet	Bit								Description
	8	7	6	5	4	3	2	1	
1	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}	Number of 1/32 ms
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}	

Octet	Bit								Description
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	
SN: 0=data is positive; 1=data is negative and in two's complement representation.									

D.5.7 Coding of PACKED_ASCII (6-BIT ASCII) DATA FORMAT

Some of the alpha-numeric data passed by the protocol is transmitted to and from the devices in the PACKED_ASCII format. PACKED_ASCII is a subset of ASCII produced by removing the two most significant bits of each ASCII character. This allows four PACKED_ASCII characters to be placed in the space of three ASCII characters. Typically, PACKED_ASCII strings are defined in even multiples of three octets, four ASCII characters.

Construction of PACKED_ASCII characters:

- Truncate Bit #6 and #7 of each ASCII character
- Pack four, 6 bit-ASCII characters into three octets

Reconstruction of ASCII characters:

- Unpack the four, 6-bit ASCII characters
- Place the complement of Bit #5 of each unpacked, 6-bit ASCII character into Bit #6
- Set Bit #7 of each of the unpacked ASCII characters to zero.

Table D.17 shows the coding of the PACKED_ASCII characters.

Table D.17 – 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

Annex E (informative)

Historical background

This standard contains the EDDL specified in IEC 61804-2:2004 and the enhancements defined in the DD-Cooperation⁷ document FF-644: EDDL Specification Revision 1.2 with the phase 1 enhancements. Subclauses 4.1 and 4.2, and Clauses 5, 6, 7, and 8 of IEC 61804-2:2004, dealing with FB, are unchanged in the second edition of IEC 61804-2; Clause 3 is now split into two clauses covering terms and definitions (new version of Clause 3) and conventions (new Clause 4).

Compared to IEC 61804-2:2004, this standard includes the following enhancements:

- Charting – Enables graphical display of real-time (actual) data from device
 - New CHART construct to define display characteristics
 - New SOURCE construct enables multiples curves on a CHART
 - New AXIS construct
- Graphing – Enables graphical display of static Y-t and XY data
 - New GRAPH construct to define display characteristics
 - New WAVEFORM construct enables multiple curves on a GRAPH.
 - New AXIS construct
- Improved Data Storage – Enables DD Developer to securely store data on the host
 - New FILE construct describes parameters that will be stored
 - New LIST construct is used with FILE to access specific parameters
- Improved User Interface (UI) – DD Developer can describe screen layout
 - Enhanced MENU construct with screen layout attributes (for example, dialog boxes)
- New METHOD builtins – support graphing, charting, storage and UI enhancements
 - New PlotCreate creates a graph or chart that can then be manipulated
 - New PlotDisplay displays a graph or chart
 - New PlotEdit allows the user to edit a graph or chart
 - New PlotRefresh causes the plot to be re-displayed
 - New PlotDestroy deletes a graph or chart
 - New ListInsert inserts an element into a list
 - New ListDeleteElementAt deletes an element from a list

EDDL is the most important and widely used digital communication descriptive language in the manufacturing and process industries.

⁷ DD-cooperation is a cooperation among the consortia of Fieldbus Foundation, Hart Communication Foundation, PROFIBUS Nutzerorganisation e.V. and, since 2006, the OPC Foundation. This second edition of EDDL extends the capabilities of the Device Description Language (EDDL) of IEC 61804-2:2004 to provide an industry standard solution for advanced visualization of intelligent device information to maintain the proven integrity of existing EDD technology across all three communication technologies according to IEC 61784-1 CPF 1, CP 3/1 and CP 3/2, and CPF 9.

Bibliography

IEC 60050-351:1998, *International Electrotechnical Vocabulary (IEV) – Part 351: Automatic control*

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

ISO 15745-1, *Industrial automation systems and integration – Open systems application integration framework – Part 1: Generic reference description*

ISO 2382 (all parts), *Information technology – Vocabulary*

ISO/AFNOR Dictionary of Computer Science

Withdrawing
IECNORM.COM : Click to view the full PDF of IEC 61804-3:2006

SOMMAIRE

AVANT-PROPOS	392
INTRODUCTION	394
1 Domaine d'application	395
2 Références normatives	395
3 Termes, définitions, termes abrégés et acronymes	396
4 Déclaration de conformité	403
5 Conventions pour les structures lexicales	403
6 Modèle EDD et EDDL	404
6.1 Présentation des EDD et d'EDDL	404
6.2 Architecture EDD	404
6.3 Concepts d'EDD	405
6.4 Principes du processus de développement d'EDD	405
6.5 Interrelations entre la structure lexicale et les définitions formelles	407
6.6 Builtins	407
6.7 Profils	407
7 Langage de description de dispositif électronique	407
7.1 Vue d'ensemble	407
7.2 Informations d'identification d'une EDD	417
7.3 AXIS	420
7.4 BLOCK	421
7.5 CHART	431
7.6 COLLECTION	433
7.7 COMMAND	435
7.8 CONNECTION	441
7.9 DOMAIN	441
7.10 EDIT_DISPLAY	442
7.11 FILE	444
7.12 GRAPH	445
7.13 GRID	447
7.14 IMAGE	450
7.15 IMPORT	451
7.16 LIKE	464
7.17 LIST	464
7.18 MENU	466
7.19 METHOD	477
7.20 PROGRAM	479
7.21 RECORD	480
7.22 REFERENCE_ARRAY	481
7.23 Relations	482
7.24 RESPONSE_CODES	483
7.25 SOURCE	484
7.26 VALUE_ARRAY	487
7.27 VARIABLE	488
7.28 VARIABLE_LIST	504

7.29 WAVEFORM	505
7.30 Attributs communs.....	512
7.31 Redirection de la sortie (OPEN et CLOSE)	515
7.32 Expression conditionnelle.....	515
7.33 Référencement.....	517
7.34 Chaînes.....	522
7.35 Expression	525
7.36 Dictionnaire de texte	530
 Annexe A (normative) Définition formelle d'EDDL	532
Annexe B (normative) Bibliothèque EDDL Builtin	611
Annexe C (informative) Exemple d'EDD	728
Annexe D (normative) Profils d'EDDL et Builtins	742
Annexe E (informative) Rappel historique	769
 Bibliographie.....	771
 Figure 1 – Position de la série CEI 61804 par rapport aux autres normes et produits	394
Figure 2 – Processus de génération d'EDD.....	406
Figure 3 – BLOCK_A	409
Figure 4 – CHART	409
Figure 5 – COLLECTION	410
Figure 6 – COMMAND	410
Figure 7 – DOMAIN.....	410
Figure 8 – EDIT_DISPLAY	411
Figure 9 – FILE.....	411
Figure 10 – GRAPH.....	411
Figure 11 – GRID.....	411
Figure 12 – IMAGE.....	412
Figure 13 – LIKE.....	412
Figure 14 – LIST.....	412
Figure 15 – MENU.....	413
Figure 16 – PROGRAM.....	413
Figure 17 – RECORD.....	413
Figure 18 – REFERENCE_ARRAY.....	414
Figure 19 – REFRESH	414
Figure 20 – UNIT	414
Figure 21 – WRITE_AS_ONE.....	414
Figure 22 – SOURCE.....	415
Figure 23 – VALUE_ARRAY	415
Figure 24 – VARIABLE	415
Figure 25 – VARIABLE_LIST	415
Figure 26 – WAVEFORM	416
Figure 27 – Mécanismes d'importation EDDL.....	452

Figure 28 – Activation de MENU (ACCESS OFFLINE)	474
Figure 29 – Action réalisée après l'entrée d'une nouvelle valeur	474
Figure 30 – Action réalisée après que toutes les entrées de VARIABLE du MENU ont été acceptées (ACCESS OFFLINE)	474
Figure 31 – Exécution de la méthode	475
Figure 32 – Activation de MENU (ACCESS ONLINE)	476
Figure 33 – Lecture cyclique des VARIABLE dynamiques (ACCESS ONLINE)	476
Figure 34 – Action réalisée après que toutes les entrées de VARIABLE du MENU sont acceptées (ACCESS ONLINE)	477
Figure 35 – Délai d'attente d'opération de lecture/écriture	503
Figure C.1 – Exemple d'écran d'opérateur utilisant l'EDD	728
Tableau 1 – Descriptions d'attribut de champ	403
Tableau 2 – Attribut DD_REVISION	417
Tableau 3 – Attribut DEVICE_REVISION	418
Tableau 4 – Attribut DEVICE_TYPE	418
Tableau 5 – Attribut EDD_PROFILE	418
Tableau 6 – Attribut EDD_VERSION	419
Tableau 7 – Attribut MANUFACTURER	419
Tableau 8 – Attribut de MANUFACTURER_EXT	420
Tableau 9 – Attributs d'AXIS	420
Tableau 10 – Attribut de MAX_VALUE, MIN_VALUE	420
Tableau 11 – Attribut de SCALING	421
Tableau 12 – Attributs de BLOCK_A	422
Tableau 13 – Attribut de CHARACTERISTIC	423
Tableau 14 – Attributs de PARAMETER	423
Tableau 15 – Attribut d'AXIS_ITEMS	423
Tableau 16 – Attribut de CHART_ITEMS	424
Tableau 17 – Attribut de COLLECTION_ITEMS	424
Tableau 18 – Attribut d'EDIT_DISPLAY_ITEMS	424
Tableau 19 – Attribut de FILE_ITEMS	425
Tableau 20 – Attribut de GRAPH_ITEMS	425
Tableau 21 – Attributs de GRID_ITEMS	425
Tableau 22 – Attribut d'IMAGE_ITEMS	425
Tableau 23 – Attribut de LIST_ITEMS	426
Tableau 24 – Attribut de MENU_ITEMS	426
Tableau 25 – Attribut de METHOD_ITEMS	427
Tableau 26 – Attributs de PARAMETER_LISTS	427
Tableau 27 – Attribut de REFERENCE_ARRAY_ITEMS	427
Tableau 28 – Attribut de REFRESH_ITEMS	428
Tableau 29 – Attribut de SOURCE_ITEMS	428
Tableau 30 – Attribut de UNIT_ITEMS	428
Tableau 31 – Attribut de WAVEFORM_ITEMS	428
Tableau 32 – Attribut de WRITE_AS_ONE_ITEMS	429

Tableau 33 – Attributs de BLOCK_B	429
Tableau 34 – Attributs de NUMBER	430
Tableau 35 – Attributs de TYPE	430
Tableau 36 – Attributs de CHART	431
Tableau 37 – Attribut de CYCLE_TIME	431
Tableau 38 – Attributs de HEIGHT/WIDTH	432
Tableau 39 – Attribut de LENGTH	433
Tableau 40 – Attribut de TYPE	433
Tableau 41 – Attributs de COLLECTION	434
Tableau 42 – item-type	434
Tableau 43 – Attributs de COMMAND	435
Tableau 44 – Attributs d'OPERATION	436
Tableau 45 – Attributs de TRANSACTION	437
Tableau 46 – Attributs de REPLY et REQUEST	437
Tableau 47 – Attribut d'INDEX	438
Tableau 48 – Attribut de BLOCK_B	439
Tableau 49 – Attribut de NUMBER	439
Tableau 50 – Attribut de SLOT	439
Tableau 51 – Attribut de CONNECTION	440
Tableau 52 – Attribut HEADER	440
Tableau 53 – Attribut de MODULE	440
Tableau 54 – Attribut de CONNECTION	441
Tableau 55 – Attributs d'APPINSTANCE	441
Tableau 56 – Attributs de DOMAIN	442
Tableau 57 – Attribut de HANDLING	442
Tableau 58 – Attributs d'EDIT_DISPLAY	443
Tableau 59 – Attribut d'EDIT_ITEMS	443
Tableau 60 – Attributs de DISPLAY_ITEM	444
Tableau 61 – Attribut de POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS	444
Tableau 62 – Attributs de FILE	445
Tableau 64 – Attribut de HEIGHT/WIDTHs	446
Tableau 65 – Attribut de CYCLE_TIME	446
Tableau 66 – Attribut de X_AXIS	447
Tableau 67 – Attributs de GRID	447
Tableau 68 – Attributs de VECTORS	448
Tableau 69 – Attribut de HANDLING	448
Tableau 70 – Attributs de HEIGHT/WIDTH	449
Tableau 71 – Attribut d'ORIENTATION	449
Tableau 72 – Attributs de VALIDITY	450
Tableau 73 – Attributs d'IMAGE	450
Tableau 74 – Attribut de PATH	450
Tableau 75 – Attribut de LINK	451
Tableau 76 – Attributs de VALIDITY	451

Tableau 77 – Description d'importation d'un appareil	453
Tableau 78 – Attributs de redéfinition	454
Tableau 79 – Règles de redéfinition pour les attributs d'AXIS	455
Tableau 80 – Règles de redéfinition pour les attributs de BLOCK_A	455
Tableau 81 – Règles de redéfinition pour les attributs de BLOCK_B	456
Tableau 82 – Règles de redéfinition pour les attributs de CHART	456
Tableau 83 – Règles de redéfinition pour les attributs de COLLECTION	456
Tableau 84 – Règles de redéfinition pour les attributs de COMMAND	457
Tableau 85 – Règles de redéfinition pour les attributs de CONNECTION	457
Tableau 86 – Règles de redéfinition pour les attributs de DOMAIN	457
Tableau 87 – Règles de redéfinition pour les attributs d'EDIT_DISPLAY	458
Tableau 88 – Règles de redéfinition pour les attributs de FILE	458
Tableau 89 – Règles de redéfinition pour les attributs de GRAPH	458
Tableau 90 – Règles de redéfinition pour les attributs de GRID	459
Tableau 91 – Règles de redéfinition pour les attributs d'IMAGE	459
Tableau 92 – Règles de redéfinition pour les attributs de LIST	459
Tableau 93 – Règles de redéfinition pour les attributs de MENU	460
Tableau 95 – Règles de redéfinition pour les attributs de PROGRAM	460
Tableau 96 – Règles de redéfinition pour les attributs de RECORD	461
Tableau 97 – Règles de redéfinition pour les attributs de REFERENCE_ARRAY	461
Tableau 98 – Règles de redéfinition pour les attributs de RESPONSE_CODES	461
Tableau 99 – Règles de redéfinition pour les attributs de SOURCE	462
Tableau 100 – Règles de redéfinition pour les attributs de VALUE_ARRAY	462
Tableau 101 – Règles de redéfinition pour les attributs de VARIABLE	462
Tableau 102 – Règles de redéfinition pour les attributs de VARIABLE_LIST	463
Tableau 103 – Règles de redéfinition pour les attributs de WAVEFORM	464
Tableau 104 – Attributs de LIKE	464
Tableau 105 – Attributs de LIST	465
Tableau 106 – Attribut de TYPE	465
Tableau 107 – Attribut de CAPACITY, COUNT	465
Tableau 109 – Attributs d'ITEMS	468
Tableau 110 – Attribut d'ACCESS	468
Tableau 111 – Attribut d'ENTRY	469
Tableau 112 – Attributs de POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS	469
Tableau 113 – Attribut de PURPOSE	471
Tableau 114 – Attribut de ROLE	472
Tableau 115 – Attribut de STYLE	472
Tableau 116 – Attributs de VALIDITY	473
Tableau 117 – Attributs METHOD	477
Tableau 118 – Attributs d'ACCESS	478
Tableau 119 – Attributs de TYPE	479
Tableau 120 – Attributs de VALIDITY	479

Tableau 121 – Attribut de PROGRAM	479
Tableau 122 – Attribut ARGUMENT	480
Tableau 123 – Attributs de RECORD	480
Tableau 124 – Attribut de REFERENCE_ARRAY	481
Tableau 125 – Attributs d'ELEMENTS	481
Tableau 126 – Attributs de REFRESH	482
Tableau 127 – Attributs d'UNIT	482
Tableau 128 – Attribut de WRITE_AS_ONE	483
Tableau 129 – Attributs de RESPONSE_CODES	483
Tableau 130 – Attributs de SOURCE	484
Tableau 131 – Attribut d'EMPHASIS	485
Tableau 132 – Attribut de LINE_COLOR	485
Tableau 133 – Attributs de LINE_TYPE	485
Tableau 134 – Attribut d'Y_AXIS	486
Tableau 135 – Attributs de VALUE_ARRAY	487
Tableau 136 – Attributs de NUMBER_OF_ELEMENTS	487
Tableau 137 – Attribut de TYPE	488
Tableau 138 – Attributs de VARIABLE	488
Tableau 139 – Attributs de CLASS	489
Tableau 140 – Attributs de TYPE	490
Tableau 141 – Attributs de DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER	492
Tableau 142 – Attributs de BIT_ENUMERATED	495
Tableau 143 – Attributs de status-class	495
Tableau 144 – Attributs ALL, AO, DV, TV	496
Tableau 145 – Attributs de types énumérés	497
Tableau 146 – Attributs de TYPE INDEX	497
Tableau 147 – Attribut du type de référence d'objet	498
Tableau 148 – Attributs de DEFAULT_REFERENCE	498
Tableau 149 – Attributs de types de chaîne	499
Tableau 150 – Attribut CONSTANT_UNIT	500
Tableau 151 – Attribut de DEFAULT_VALUE	500
Tableau 152 – Attribut de HANDLING	500
Tableau 153 – Attribut d'INITIAL_VALUE	501
Tableau 154 – Attributs de POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS	501
Tableau 155 – Attributs de READ/WRITE_TIMEOUT	503
Tableau 156 – Attribut de STYLE	504
Tableau 157 – Attributs de VALIDITY	504
Tableau 158 – Attributs de VARIABLE_LIST	504
Tableau 159 – Attributs de WAVEFORM	505
Tableau 160 – Attribut de TYPE	505
Tableau 161 – Attribut de XY	506
Tableau 162 – Attribut de YT	506

Tableau 163 – Attribut de HORIZONTAL.....	507
Tableau 164 – Attribut de VERTICAL.....	508
Tableau 165 – Attribut d'EMPHASIS.....	508
Tableau 166 – Attribut de HANDLING.....	509
Tableau 167 – Attribut EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS.....	509
Tableau 168 – Attribut de KEY_POINTS.....	509
Tableau 169 – Attribut de X_VALUES, Y_VALUES.....	510
Tableau 170 – Attribut de LINE_COLOR.....	511
Tableau 171 – Attributs de LINE_TYPE.....	511
Tableau 172 – Attribut d'Y_AXIS.....	512
Tableau 173 – Attribut de DEFINITION.....	512
Tableau 174 – Attribut HELP.....	513
Tableau 175 – Attribut LABEL.....	513
Tableau 176 – Attributs de MEMBERS.....	514
Tableau 177 – Attribut de RESPONSE_CODES.....	514
Tableau 178 – Attributs d'OPEN et de CLOSE.....	515
Tableau 179 – Instruction conditionnelle IF, SELECT.....	516
Tableau 180 – Référencement d'une instance d'EDD.....	517
Tableau 181 – Référencement d'éléments de VARIABLE.....	517
Tableau 182 – Référencement d'éléments de RECORD.....	517
Tableau 183 – Référencement d'éléments de VALUE_ARRAY.....	518
Tableau 184 – Référencement des membres de COLLECTION.....	518
Tableau 185 – Référencement des membres de REFERENCE_ARRAY.....	518
Tableau 186 – Référencement des membres de VARIABLE_LISTS.....	519
Tableau 187 – Référencement des membres de BLOCK_A PARAMETERS.....	519
Tableau 188 – Référencement des membres de BLOCK_A PARAMETER_LISTS.....	519
Tableau 189 – Référencement des membres de BLOCK_A LOCAL_PARAMETER.....	520
Tableau 190 – Référencement de BLOCK_A CHARACTERISTICS.....	520
Tableau 191 – Référencement des membres de FILE.....	520
Tableau 192 – Référencement d'éléments de LIST.....	521
Tableau 193 – Référencement des membres de CHART.....	521
Tableau 194 – Référencement des membres de GRAPH.....	521
Tableau 195 – Référencement des membres de SOURCE.....	522
Tableau 196 – Référencement d'AXIS d'un GRAPH. SOURCE, WAVEFORM.....	522
Tableau 197 – Chaîne sous la forme d'un littéral chaîne.....	522
Tableau 198 – Chaîne sous la forme d'une variable chaîne.....	523
Tableau 199 – Chaîne en tant que valeur d'énumération.....	523
Tableau 200 – Chaîne en tant que référence de dictionnaire.....	523
Tableau 201 – Référencement d'attributs HELP et LABEL d'instances d'EDD.....	524
Tableau 202 – Opération de chaîne.....	524
Tableau 203 – Spécificateur de format.....	524
Tableau 204 – Expressions primaires.....	525
Tableau 205 – Valeurs d'attribut de VARIABLE.....	526

Tableau 206 – Valeurs d'attribut d'AXIS	526
Tableau 207 – Valeurs d'attribut de LIST	526
Tableau 208 – Expressions unaires	527
Tableau 209 – Opérateurs multiplicatifs	528
Tableau 210 – Opérateurs additifs	528
Tableau 211 – Opérateurs de décalage	528
Tableau 212 – Opérateurs relationnels	529
Tableau 213 – Opérateurs d'égalité	529
Tableau 214 – Attributs de dictionnaire de texte	531
Tableau A.1 – Conventions pour les constantes entières	536
Tableau A.2 – Utilisation des séquences d'échappement dans les littéraux chaînes	537
Tableau A.3 – Utilisation des codes de langue dans les littéraux chaînes	537
Tableau A.4 – Opérateurs EDDL	538
Tableau A.5 – Mots clés EDDL	538
Tableau B.1 – Format des tables d'éléments lexicaux de Builtins	611
Tableau B.2 – Contenu de la table d'éléments lexicaux	611
Tableau B.3 – Builtin abort	612
Tableau B.4 – Builtin abort_on_all_comm_errors	612
Tableau B.5 – Builtin ABORT_ON_ALL_COMM_STATUS	613
Tableau B.6 – Builtin ABORT_ON_ALL_DEVICE_STATUS	613
Tableau B.7 – Builtin ABORT_ON_ALL_RESPONSE_CODES	614
Tableau B.8 – Builtin abort_on_all_response_codes	614
Tableau B.9 – Builtin abort_on_comm_error	615
Tableau B.10 – Builtin ABORT_ON_COMM_ERROR	615
Tableau B.11 – Builtin ABORT_ON_COMM_STATUS	616
Tableau B.12 – Builtin ABORT_ON_DEVICE_STATUS	616
Tableau B.13 – Builtin ABORT_ON_NO_DEVICE	617
Tableau B.14 – Builtin ABORT_ON_RESPONSE_CODE	618
Tableau B.15 – Builtin abort_on_response_code	618
Tableau B.16 – Builtin abs	618
Tableau B.17 – Builtin ACKNOWLEDGE	619
Tableau B.18 – Builtin acknowledge	619
Tableau B.19 – Builtin acos	619
Tableau B.20 – Builtin add_abort_method	620
Tableau B.21 – Builtin add_abort_method	621
Tableau B.22 – Builtin asin	621
Tableau B.23 – Builtin assign	621
Tableau B.24 – Builtin assign_double	622
Tableau B.25 – Builtin assign_float	622
Tableau B.26 – Builtin assign_int	623
Tableau B.27 – Builtin assign_var	623
Tableau B.28 – Builtin atan	623
Tableau B.29 – Builtin ATOF	624

Tableau B.30 – Builtin atof.....	624
Tableau B.31 – Builtin ATOI.....	624
Tableau B.32 – Builtin atoi.....	624
Tableau B.33 – Builtin BUILD_MESSAGE.....	625
Tableau B.34 – Builtin cbrt.....	625
Tableau B.35 – Builtin ceil.....	625
Tableau B.36 – Builtin cos.....	626
Tableau B.37 – Builtin cosh.....	626
Tableau B.38 – Builtin dassign.....	626
Tableau B.39 – Builtin Date_to_DayOfMonth.....	627
Tableau B.40 – Builtin Date_to_Month.....	627
Tableau B.41 – Builtin Date_to_Year.....	628
Tableau B.42 – Builtin DELAY.....	628
Tableau B.43 – Builtin delay.....	629
Tableau B.44 – Builtin DELAY_TIME.....	629
Tableau B.45 – Builtin delayfor.....	629
Tableau B.46 – Builtin DICT_ID.....	630
Tableau B.47 – Builtin discard_on_exit.....	631
Tableau B.48 – Builtin display.....	631
Tableau B.49 – Builtin display_builtin_error.....	631
Tableau B.50 – Builtin display_comm_error.....	632
Tableau B.51 – Builtin display_comm_status.....	632
Tableau B.52 – Builtin display_device_status.....	633
Tableau B.53 – Builtin display_dynamics.....	633
Tableau B.54 – Builtin display_message.....	634
Tableau B.55 – Builtin display_response_code.....	635
Tableau B.56 – Builtin display_response_status.....	635
Tableau B.57 – Builtin display_xmtr_status.....	636
Tableau B.58 – Builtin edit_device_value.....	636
Tableau B.59 – Builtin edit_local_value.....	637
Tableau B.60 – Builtin exp.....	637
Tableau B.61 – Builtin ext_send_command.....	638
Tableau B.62 – Builtin ext_send_command_trans.....	639
Tableau B.63 – Builtin fail_on_all_comm_errors.....	639
Tableau B.64 – Builtin fail_on_all_response_codes.....	640
Tableau B.65 – Builtin fail_on_comm_error.....	640
Tableau B.66 – Builtin fail_on_response_code.....	641
Tableau B.67 – Builtin fassign.....	641
Tableau B.68 – Builtin fgetval.....	642
Tableau B.69 – Builtin float_value.....	642
Tableau B.70 – Builtin floor.....	642
Tableau B.71 – Builtin fmod.....	643
Tableau B.72 – Builtin fsetval.....	643

Tableau B.73 – Builtin ftoa.....	644
Tableau B.74 – Builtin fvar_value.....	644
Tableau B.75 – Builtin get_acknowledgement.....	644
Tableau B.76 – Builtin get_comm_error.....	645
Tableau B.77 – Builtin get_comm_error_string.....	646
Tableau B.78 – Builtin get_date.....	646
Tableau B.79 – Builtin get_date_value.....	647
Tableau B.80 – Builtin get_dds_error.....	647
Tableau B.81 – Builtin GET_DEV_VAR_VALUE.....	648
Tableau B.82 – Builtin get_dev_var_value.....	649
Tableau B.83 – Builtin get_dictionary_string.....	649
Tableau B.84 – Builtin get_double.....	650
Tableau B.85 – Builtin get_double_value.....	650
Tableau B.86 – Builtin get_enum_string.....	651
Tableau B.87 – Builtin get_float.....	651
Tableau B.88 – Builtin get_float_value.....	652
Tableau B.89 – Builtin GET_LOCAL_VAR_VALUE.....	652
Tableau B.90 – Builtin get_local_var_value.....	653
Tableau B.91 – Builtin get_more_status.....	653
Tableau B.92 – Builtin get_resolve_status.....	654
Tableau B.93 – Builtin get_response_code.....	655
Tableau B.94 – Builtin get_response_code_string.....	655
Tableau B.95 – Builtin get_signed.....	656
Tableau B.96 – Builtin get_signed_value.....	657
Tableau B.97 – Builtin get_status_code_string.....	657
Tableau B.98 – Builtin get_status_string.....	658
Tableau B.99 – Builtin get_stddict_string.....	658
Tableau B.100 – Builtin get_string.....	659
Tableau B.101 – Builtin get_string_value.....	659
Tableau B.102 – Builtin GET_TICK_COUNT.....	660
Tableau B.103 – Builtin get_unsigned.....	660
Tableau B.104 – Builtin get_unsigned_value.....	661
Tableau B.105 – Builtin iassign.....	661
Tableau B.106 – Builtin igetval.....	662
Tableau B.107 – Builtin IGNORE_ALL_COMM_STATUS.....	663
Tableau B.108 – Builtin IGNORE_ALL_DEVICE_STATUS.....	663
Tableau B.110 – Builtin IGNORE_COMM_ERROR.....	664
Tableau B.111 – Builtin IGNORE_COMM_STATUS.....	665
Tableau B.112 – Builtin IGNORE_DEVICE_STATUS.....	665
Tableau B.113 – Builtin IGNORE_NO_DEVICE.....	666
Tableau B.114 – Builtin IGNORE_RESPONSE_CODE.....	666
Tableau B.115 – Builtin int_value.....	667
Tableau B.116 – Builtin is_NaN.....	667

Tableau B.117 – Builtin isetval.....	668
Tableau B.118 – Builtin ITEM_ID	668
Tableau B.119 – Builtin ITOA.....	668
Tableau B.120 – Builtin itoa	669
Tableau B.121 – Builtin ivar_value	669
Tableau B.122 – Builtin lassign	669
Tableau B.123 – Builtin lgetval.....	670
Tableau B.124 – Builtin ListDeleteElementAt	670
Tableau B.126 – Builtin log	671
Tableau B.127 – Builtin log10	671
Tableau B.128 – Builtin log2	672
Tableau B.129 – Éléments lexicaux du Builtin LOG_MESSAGE	672
Tableau B.130 – Builtin long_value	673
Tableau B.131 – Builtin lsetval.....	673
Tableau B.132 – Builtin lvar_value	673
Tableau B.133 – Builtin MEMBER_ID.....	674
Tableau B.134 – Builtin MenuDisplay	674
Tableau B.135 – Builtin method_abort.....	675
Tableau B.136 – Builtin process_abort.....	675
Tableau B.137 – Builtin pow	676
Tableau B.138 – Builtin process_abort.....	676
Tableau B.139 – Builtin put_date	676
Tableau B.140 – Builtin put_date_value	677
Tableau B.141 – Builtin put_double.....	678
Tableau B.142 – Builtin put_double_value	678
Tableau B.143 – Builtin put_float	679
Tableau B.144 – Builtin put_float_value	679
Tableau B.145 – Builtin PUT_MESSAGE	680
Tableau B.146 – Builtin put_message	680
Tableau B.147 – Builtin put_signed.....	681
Tableau B.148 – Builtin put_signed_value.....	681
Tableau B.149 – Builtin put_string	682
Tableau B.150 – Builtin put_string_value	683
Tableau B.151 – Builtin put_unsigned	684
Tableau B.152 – Builtin put_unsigned_value.....	684
Tableau B.153 – Éléments lexicaux du Builtin READ_COMMAND.....	685
Tableau B.154 – Builtin read_value.....	685
Tableau B.155 – Builtin remove_abort_method	686
Tableau B.156 – Builtin remove_abort_method	686
Tableau B.157 – Builtin remove_all_abort_methods.....	687
Tableau B.158 – Builtin resolve_array_ref.....	687
Tableau B.159 – Builtin resolve_block_ref	688
Tableau B.160 – Builtin resolve_param_list_ref	688

Tableau B.161 – Builtin resolve_parm_ref.....	689
Tableau B.162 – Builtin resolve_record_ref.....	690
Tableau B.163 – Builtin retry_on_all_comm_errors	690
Tableau B.164 – Builtin RETRY_ON_ALL_COMM_STATUS	691
Tableau B.165 – Builtin RETRY_ON_ALL_DEVICE_STATUS	691
Tableau B.166 – Builtin RETRY_ON_ALL_RESPONSE_CODES.....	692
Tableau B.167 – Builtin retry_on_all_response_codes	692
Tableau B.168 – Builtin RETRY_ON_COMM_ERROR.....	693
Tableau B.169 – Builtin retry_on_comm_error	693
Tableau B.170 – Builtin RETRY_ON_COMM_STATUS	694
Tableau B.172 – Builtin RETRY_ON_NO_DEVICE.....	695
Tableau B.173 – Builtin RETRY_ON_RESPONSE_CODE.....	696
Tableau B.174 – Builtin retry_on_response_code	696
Tableau B.175 – Builtin round.....	697
Tableau B.176 – Builtin rspcode_string.....	697
Tableau B.177 – Builtin save_on_exit	698
Tableau B.178 – Builtin save_values	698
Tableau B.179 – Builtin SELECT_FROM_LIST	699
Tableau B.180 – Builtin select_from_list	699
Tableau B.181 – Builtin select_from_menu	700
Tableau B.182 – Builtin send	701
Tableau B.183 – Builtin send_all_values.....	701
Tableau B.184 – Builtin send_command	702
Tableau B.185 – Builtin send_command_trans.....	702
Tableau B.186 – Builtin send_on_exit	703
Tableau B.187 – Builtin send_trans.....	704
Tableau B.188 – Builtin send_value.....	704
Tableau B.189 – Builtin SET_NUMBER_OF_RETRIES	705
Tableau B.190 – Builtin sin	705
Tableau B.191 – Builtin sinh	705
Tableau B.192 – Builtin sqrt.....	706
Tableau B.193 – Builtin strcmp	706
Tableau B.194 – Builtin strlen	706
Tableau B.195 – Builtin strlwr	707
Tableau B.196 – Builtin strmid	707
Tableau B.197 – Builtin strstr.....	708
Tableau B.198 – Builtin strtrim	708
Tableau B.199 – Builtinstrupr.....	708
Tableau B.200 – Builtin tan	709
Tableau B.201 – Builtin tanh	709
Tableau B.202 – Builtin To_Date_and_Time	709
Tableau B.203 – Builtin trunc	710
Tableau B.204 – Builtin VARID	710

Tableau B.205 – Builtin vassign	711
Tableau B.206 – Builtin WRITE_COMMAND	711
Tableau B.207 – Builtin XMTR_ABORT_ON_ALL_COMM_STATUS	712
Tableau B.208 – Builtin XMTR_ABORT_ON_ALL_DEVICE_STATUS	712
Tableau B.209 – Builtin XMTR_ABORT_ON_ALL_RESPONSE_CODES	713
Tableau B.210 – Builtin XMTR_ABORT_ON_COMM_ERROR	713
Tableau B.211 – Builtin XMTR_ABORT_ON_COMM_STATUS	714
Tableau B.212 – Builtin XMTR_ABORT_ON_DATA	714
Tableau B.213 – Builtin XMTR_ABORT_ON_DEVICE_STATUS	715
Tableau B.214 – Builtin XMTR_ABORT_ON_NO_DEVICE	715
Tableau B.215 – Builtin XMTR_ABORT_ON_RESPONSE_CODE	716
Tableau B.216 – Builtin XMTR_IGNORE_ALL_COMM_STATUS	717
Tableau B.217 – Builtin XMTR_IGNORE_ALL_DEVICE_STATUS	717
Tableau B.218 – Builtin XMTR_IGNORE_ALL_RESPONSE_CODES	718
Tableau B.219 – Builtin XMTR_IGNORE_COMM_ERROR	718
Tableau B.220 – Builtin XMTR_IGNORE_COMM_STATUS	719
Tableau B.221 – Builtin XMTR_IGNORE_DEVICE_STATUS	719
Tableau B.222 – Builtin XMTR_IGNORE_NO_DEVICE	720
Tableau B.223 – Builtin XMTR_IGNORE_RESPONSE_CODE	720
Tableau B.224 – Builtin XMTR_RETRY_ON_ALL_DEVICE_STATUS	721
Tableau B.225 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODE	722
Tableau B.226 – Builtin XMTR_RETRY_ON_ALL_RESPONSE_CODES	722
Tableau B.227 – Builtin XMTR_RETRY_ON_COMM_ERROR	723
Tableau B.228 – Builtin XMTR_RETRY_ON_COMM_STATUS	723
Tableau B.229 – Builtin XMTR_RETRY_ON_DATA	724
Tableau B.230 – Builtin XMTR_RETRY_ON_DEVICE_STATUS	724
Tableau B.231 – Builtin XMTR_RETRY_ON_NO_DEVICE	725
Tableau B.232 – Builtin XMTR_RETRY_ON_RESPONSE_CODE	725
Tableau B.233 – Builtin YearMonthDay_to_Date	726
Tableau B.234 – Contenu des tableaux de description de codes de retour	726
Tableau B.235 – Description de code de retour	727
Tableau D.1 – Tables de sélection de profil	742
Tableau D.2 – Tables de profil de définition formelle EDDL	742
Tableau D.3 – Contenu des tables de sélection	742
Tableau D.4 – Sélection d'éléments EDDL pour PROFIBUS	743
Tableau D.5 – Profil de Builtin pour PROFIBUS	746
Tableau D.6 – Sélection d'éléments EDDL pour Fieldbus Foundation	749
Tableau D.7 – Profil de Builtin pour Fieldbus Foundation	753
Tableau D.8 – Sélection d'éléments EDDL pour HCF	757
Tableau D.9 – Profil de Builtin pour HCF	760
Tableau D.10 – Types de données METHOD DEFINITIONS	764
Tableau D.11 – VARIABLE TYPES	764
Tableau D.12 – Codage de DATE	765

Tableau D.13 – Codage de DATE_AND_TIME	766
Tableau D.14 – Codage de DURATION	766
Tableau D.15 – Codage de TIME	766
Tableau D.16 – Codage de TIME_VALUE	767
Tableau D.17 – Codage de PACKED_ASCII	768

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2006

Withd2Wm

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

BLOCS FONCTIONNELS (FB) POUR LES PROCÉDÉS INDUSTRIELS –**Partie 3: Langage de description électronique de produit (EDDL)****AVANT-PROPOS**

- 1) La Commission Électrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI 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, la CEI – 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 la CEI"). 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 la CEI, participent également aux travaux. La CEI 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 la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
- 3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI 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 la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
- 5) La CEI n'a prévu aucune procédure de marquage valant indication d'approbation et n'engage pas sa responsabilité pour les équipements déclarés conformes à une de ses Publications.
- 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 la CEI, 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 la CEI, 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 la CEI ou de toute autre Publication de la CEI, 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.

La CEI ne prend pas position eu égard à la preuve, la validité et la portée de ces droits de propriété. Le détenteur de ces droits a donné l'assurance à la CEI qu'il consent à négocier des licences avec des demandeurs du monde entier, en des termes et à des conditions raisonnables et non discriminatoires. À ce propos, la déclaration du détenteur des droits au brevet est enregistrée à la CEI.

L'attention est d'autre part attirée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits au brevet autres que ceux qui ont été mentionnés ci-dessus. La CEI ne saurait être tenue pour responsable de l'identification de l'un quelconque ou de la totalité de ces droits de propriété industrielle.

La présente Norme internationale a été établie par le sous-comité 65C: Communications numériques, du comité d'études 65 de la CEI: Mesure et commande dans les processus industriels.

Cette première édition annule et remplace la spécification EDDL donnée dans la première édition de la CEI 61804-2, publiée en 2004¹.

La présente version bilingue (2012-12) correspond à la version anglaise monolingue publiée en 2006-09.

¹ Le contexte historique de la spécification EDDL est présenté à l'Annexe E.

Le texte anglais de cette norme est issu des documents 65C/406/CDV et 65C/421/RVC.

Le rapport de vote 65C/421/RVC donne toute information sur le vote ayant abouti à l'approbation de cette norme.

La version française de cette norme n'a pas été soumise au vote.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

La liste de toutes les parties de la série CEI 61804, présentées sous le titre général *Blocs de fonction (FB) pour processus et commande*, peut être consultée sur le site web de la CEI.

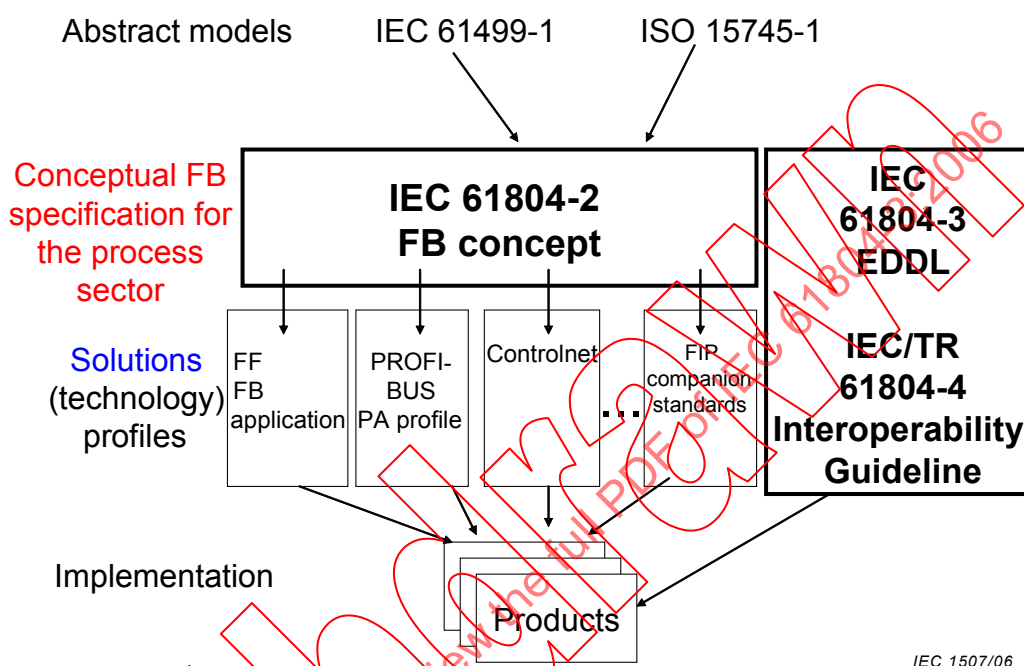
Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de maintenance indiquée sur le site web de la CEI sous "<http://webstore.iec.ch>" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite;
- supprimée;
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

Le langage EDDL établit un lien entre la spécification conceptuelle de bloc fonction de la CEI 61804-2 et une implémentation de produit. Il permet aux fabricants d'utiliser la même méthode de description pour des dispositifs basés sur différentes technologies et diverses plates-formes. La Figure 1 présente ces aspects.



IEC 1507/06

Légende

Anglais	Français
Abstract models	Modèles abstraits
IEC 61499-1	CEI 61499-1
Conceptual FB specification for the process sector	Spécification de FB conceptuelle pour le secteur de traitement
IEC 61804-2	CEI 61804-2
IEC 61804-3	CEI 61804-3
Solutions (technology profiles)	Solutions (profils technologiques)
FF FB application	Application FF FB
PA profile	Profil PA
FIP companion standards	Normes d'accompagnement FIP
Implementation	Mise en œuvre
Products	Produits
Interoperability Guideline	Lignes directrices d'interopérabilité
IEC/TR 61804-4	CEI/TR 61804-4

Figure 1 – Position de la série CEI 61804 par rapport aux autres normes et produits

BLOCS FONCTIONNELS (FB) POUR LES PROCÉDÉS INDUSTRIELS –

Partie 3: Langage de description électronique de produit (EDDL)

1 Domaine d'application

La présente partie de la série CEI 61804 spécifie la technologie de langage de description électronique d'appareil (Electronic Device Description Language: EDDL), qui permet en utilisant les outils d'ingénierie l'intégration de produit dans les systèmes tout au long du cycle de vie.

La présente norme spécifie EDDL en tant que langage générique pour décrire les propriétés des composants système mettant en œuvre des automatismes. EDDL est capable de décrire

- les paramètres des dispositifs et leurs dépendances;
- les fonctions des dispositifs, par exemple le mode de simulation, l'étalonnage;
- les représentations graphiques, par exemple les menus;
- les interactions avec les appareils de commande
- les représentations graphiques
 - interface utilisateur avancée
 - système graphique
- le répertoire des données mémorisées.

EDDL est destiné à être utilisé pour créer une description d'un appareil électronique (Electronic Device Description: EDD). Cette EDD est utilisée par des outils appropriés pour générer un code interprété qui prend en charge la manipulation des paramètres, le fonctionnement et la surveillance des composants système mettant en œuvre des automatismes, tels que des E/S, des contrôleurs, des capteurs et des contrôleurs programmables. Les outils d'implémentation ne font pas partie du domaine d'application de la présente norme.

La présente norme spécifie la sémantique et la structure lexicale indépendamment de la syntaxe. Une syntaxe spécifique est définie à l'Annexe A, mais le modèle sémantique peut être utilisé avec d'autres syntaxes.

NOTE 1 Le langage EDDL peut également être utilisé pour la description des propriétés de produit pour d'autres domaines.

Le langage EDDL et l'EDD associée à un appareil est applicable à l'automatisation industrielle.

NOTE 2 L'automatisation industrielle peut comprendre des appareils tels que des modules d'entrée/sortie numériques et analogiques génériques, des contrôleurs de mouvement, des interfaces homme-machine, des capteurs, des contrôleurs en boucle fermée, des codeurs, des vannes hydrauliques, et des contrôleurs programmables.

La présente Norme internationale satisfait aux exigences de l'Article 9 de la CEI 61804-1.

2 Références normatives

Les documents de référence suivants sont indispensables pour l'application 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).

CEI 61499-1:2005, *Blocs fonctionnels – Partie 1: Architecture*

IEC 61804-1:2003, *Function blocks (FB) for process control – Part 1: Overview of system aspects*

ISO/CEI 2022, *Technologies de l'information – Structure de code de caractères et techniques d'extension*

ISO/CEI 2375:2003, *Technologies de l'information – Procédure pour l'enregistrement des séquences d'échappement et des jeux de caractères codés*

ISO/CEI 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Le modèle de base*

ISO/CEI 8859-1:1998, *Technologies de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 1: Alphabet latin no 1*

ISO/CEI 9899, *Langages de programmation – C*

ISO/CEI 10646-1:2000, *Technologies de l'information – Jeu universel de caractères codés sur plusieurs octets (JUC) – Partie 1: Architecture et plan multilingue de base*

ISO 639, *Codes pour la représentation des noms de langue*

ISO 3166, *Codes pour la représentation des noms de pays et de leurs subdivisions*

IEEE 754:1985 (R1990), *Binary Floating-Point Arithmetic*

3 Termes, définitions, termes abrégés et acronymes

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions suivants, dont certains ont été compilés à partir des documents référencés, s'appliquent.

NOTE Les termes de la CEI 60050-351 s'appliquent partiellement.

3.2

algorithme

ensemble fini de règles bien définies pour la solution d'un problème en un nombre fini d'opérations

3.3

attribut

description d'une caractéristique ou un aspect visible de l'extérieur d'un objet

NOTE Les attributs d'un objet contiennent des informations pour des parties variables d'un objet. Ils donnent typiquement des informations de statut ou régissent le fonctionnement d'un objet. Les attributs peuvent également affecter le comportement d'un objet. Les attributs sont divisés en attributs de classe et en attributs d'instance.

3.4

Builtin

sous-programme prédéfini pour la communication et l'affichage exécuté par l'application EDD

3.5**configuration** (d'un système ou d'un appareil)

étape de la conception d'un système: sélection d'unités fonctionnelles, assignation de leurs emplacements et définition de leurs interconnexions

[CEI 61499-1]

3.6**données**

représentation de faits, concepts ou instructions d'une manière formalisée adaptée pour une communication, une interprétation ou un traitement par des êtres humains ou par des moyens automatiques

[Dictionnaire de l'informatique ISO/AFNOR]

3.7**connexion de données**

association établie entre des unités fonctionnelles pour le transport de données

[CEI 61499-1]

3.8**données d'entrée**

interface d'un FB qui reçoit des données depuis une connexion de données

[CEI 61499-1]

3.9**données de sortie**

interface d'un FB, qui transmet des données à une connexion de données

[CEI 61499-1]

3.10**type de données**

ensemble de valeurs conjointement avec un ensemble d'opérations autorisées

[série ISO 2382]

3.11**appareil**

entité physique indépendante capable d'exécuter une ou plusieurs fonctions dans un contexte particulier et délimitées par ses interfaces

[CEI 61499-1]

3.12**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.

3.13**processeur EDDL**

processeur ou programme, qui traduit l'EDD en forme exécutable qui peut être traitée par une application EDD

3.14

profil EDDL

sélection des éléments pris en charge de la structure lexicale EDDL comprenant les définitions de syntaxe pour une pluralité de consortiums spécifiques

3.15

langage de description de dispositif électronique EDDL

méthodologie pour décrire un/des paramètre(s) d'un composant de système mettant en œuvre des automatismes

3.16

description de dispositif électronique EDD

collection de données contenant le(s) paramètre(s) de l'appareil, leurs dépendances, leur représentation graphique et une description des ensembles de données qui sont transférés

NOTE La description de l'appareil électronique est créée en utilisant le langage de description électronique de produit (EDDL).

3.17

source de description de l'appareil électronique EDDS

fichier ASCII contenant une description de dispositif spécifique

3.18

technologie de description de l'appareil électronique EDDT

technologie qui comprend le processus de développement d'une EDD, l'usage d'une EDD et la chaîne d'outils impliquée

3.19

vérificateur de description de l'appareil électronique

outil d'essai qui vérifie la structure lexicale et partiellement la sémantique des sources d'EDD afin de garantir la conformité de sources d'EDD avec le langage EDD

3.20

compilateur de description de l'appareil électronique

compilateur qui traduit la source d'EDD dans un format interne qui est utilisé par l'interpréteur d'EDD

3.21

interpréteur de description de dispositif électronique EDDI

interpréteur qui utilise la source d'EDD ou un format interne qui est donné par le compilateur EDDL pour fournir les informations d'EDD à l'utilisateur d'EDD

3.22

entité

élément particulier, tel qu'une personne, un emplacement, un processus, un objet, un concept, une association, ou un événement

[CEI 61499-1]

3.23

événement

occurrence instantanée qui est significative pour planifier l'exécution d'un algorithme

[CEI 61499-1]

NOTE L'exécution d'un algorithme peut utiliser des variables associées à un événement.

3.24

exception

événement qui cause la suspension d'une exécution normale

[CEI 61499-1]

3.25

fonction

fonction spécifique d'une entité ou son action caractéristique

[CEI 61499-1]

3.26

unité fonctionnelle

entité matérielle ou logicielle, ou les deux, capable de remplir une fonction spécifiée

[Dictionnaire de l'informatique ISO/AFNOR]

3.27

bloc de fonction (instance de bloc de fonction)

unité fonctionnelle logicielle comprenant une copie individuelle nommée d'une structure de données et des opérations associées spécifiées par un type de FB correspondant

[CEI 61499-1]

NOTE Les opérations classiques d'un FB comprennent la modification des valeurs des données dans sa structure de données associée.

3.28

matériel

équipement physique, par opposition à des programmes, des procédures, des règles et documentation associée

[Dictionnaire de l'informatique ISO/AFNOR]

3.29

mise en œuvre

phase de développement dans laquelle le matériel et le logiciel d'un système deviennent opérationnels

[CEI 61499-1]

3.30

variable d'entrée

variable dont la valeur est donnée par une entrée de données, et qui peut être utilisée dans une ou plusieurs opérations d'un FB

NOTE Un paramètre d'entrée d'un FB, comme défini dans la CEI 61131-3, est une variable d'entrée.

[CEI 61499-1]

3.31

instance

unité fonctionnelle comprenant une entité individuelle nommée avec les attributs d'un type défini

[CEI 61499-1]

3.32

nom d'instance

identifiant associé à, et désignant, une instance

[CEI 61499-1]

3.33

instantiation

création d'une instance d'un type spécifié

[CEI 61499-1]

3.34

interface

frontière commune entre deux unités fonctionnelles, définie par des caractéristiques fonctionnelles, des caractéristiques de signal ou d'autres caractéristiques appropriées

[CEI 60050-351:1998, 11-19]

3.35

variable interne

variable dont la valeur est utilisée ou modifiée par une ou plusieurs opérations d'un FB mais n'est pas fournie par une entrée de données ou à une sortie de données

[CEI 61499-1]

3.36

invocation

processus d'initiation de l'exécution de la séquence d'opérations spécifiée dans un algorithme

[CEI 61499-1]

3.37

mapping

ensemble de valeurs ayant une correspondance définie avec les quantités ou valeurs d'un autre ensemble

[Dictionnaire de l'informatique ISO/AFNOR]

3.38

modèle

représentation d'un processus, dispositif ou concept du monde réel

[CEI 61499-1]

3.39

opération

action bien définie qui, lorsqu'elle est appliquée à une combinaison admissible quelconque d'entités connues, produit une nouvelle entité

[Dictionnaire de l'informatique ISO/AFNOR]

3.40

variable de sortie

variable dont la valeur est établie par une ou plusieurs opérations d'un FB, et est transmise à une sortie de données

NOTE Un paramètre de sortie d'un FB, comme défini dans la CEI 61131-3, est une variable de sortie.

[CEI 61499-1]

3.41**paramètre**

variable à laquelle est donnée une valeur constante pour une application spécifiée et qui peut désigner l'application

[Dictionnaire de l'informatique ISO/AFNOR]

3.42**préprocesseur**

partie de l'environnement d'exécution qui transforme les informations fournies dans une EDD

3.43**directives de préprocesseur**

description de conditions pour filtrer le code d'EDD avant compilation ou interprétation

NOTE Par exemple, une directive de préprocesseur permettant de définir des noms pour des constantes ou écrire des macros pour rendre le code plus facile à lire.

3.44**ressource**

unité fonctionnelle contenue dans un appareil qui a un contrôle indépendant de son fonctionnement, et qui fournit différents services à des applications, comprenant la programmation et l'exécution d'algorithmes

NOTE 1 L'objet RESSOURCE défini dans la CEI 61131-3 est un élément de langage de programmation correspondant à la ressource définie ci-dessus.

NOTE 2 Un appareil contient une ou plusieurs ressources.

3.45**application de gestion de ressource**

application dont la fonction primaire est la gestion d'une ressource unique

[CEI 61499-1]

3.46**service**

capacité fonctionnelle d'une ressource, qui peut être modélisée par une séquence de primitives de service

[CEI 61499-1]

3.47**logiciel**

création intellectuelle comprenant les programmes, procédures, règles et documentation associée relatifs au fonctionnement d'un système

[CEI 61499-1]

3.48**système**

ensemble d'éléments corrélés considérés dans un contexte défini comme un ensemble et séparé de son environnement

[VEI 350-11-01:1998]

NOTE 1 De tels éléments peuvent être à la fois des objets matériels et des concepts, ainsi que les résultats de ceux-ci (des formes d'organisation, des méthodes mathématiques et des langages de programmation, par exemple).

NOTE 2 Le système est considéré comme étant séparé de l'environnement et d'autres systèmes externes par une surface imaginaire, qui peut couper les liens entre ceux-ci et le système en question.

3.49

dictionnaire de texte

collection de textes multilingues ou autres dans l'EDD

NOTE Les références dans une EDD sont utilisées pour sélectionner un dictionnaire de texte approprié.

3.50

type

élément logiciel, qui spécifie les attributs communs partagés par toutes les instances du type

[CEI 61499-1]

3.51

nom de type

identifiant associé à, et désignant, un type

[CEI 61499-1]

3.52

variable

entité logicielle qui peut prendre différentes valeurs, une à la fois

NOTE 1 Les valeurs d'une variable sont généralement restreintes à un certain type de données.

NOTE 2 Les variables sont décrites comme étant des variables d'entrée, des variables de sortie et des variables internes.

[CEI 61499-1]

3.53 Termes abrégés et acronymes

ANSI	American National Standard Institute (Institut national américain de normalisation)
ANSI C	American National Standard Institute for the programming language C (Institut national américain de normalisation pour le langage de programmation C) (voir ISO/CEI 9899)
ASCII	American Standard Code for Information Interchange (Code standard américain pour l'échange d'informations) (voir ISO/CEI 10646-1)
ASN.1	Abstract Syntax Notation 1 (Notation de syntaxe abstraite 1)
DD	Description d'équipement
EDD	Electronic Device Description (Description d'équipement électronique)
EDDI	Electronic Device Description Interpreter (Interpréteur de description d'équipement électronique)
EDDL	Electronic Device Description Language (Langage de description d'équipement électronique)
EUC	Extended Unit Code (Code d'unité étendu) (voir ISO/CEI 2022)
FB	Function Block (Bloc fonctionnel)
IHM	Interface homme machine
HTML	Hypertext Mark-up Language
E/S	Entrée/Sortie
ID	Identifiant
OSI	Open Systems Interconnection (Interconnexion de systèmes ouverts)
wao	Write as one

4 Déclaration de conformité

Une déclaration de conformité à un profil de la présente Norme internationale doit être établie² comme étant

– conforme à la CEI 61804-3:2006 Annexe D.n <Type>

ou

– conforme à la CEI 61804-3/Ed1.0 Annexe D.n <Type>

où le Type entre les crochets obliques < > est facultatif et les crochets obliques ne doivent pas être inclus.

NOTE Le Type peut être, par exemple, le nom du protocole de communication utilisé, tel que PROFIBUS, Foundation Fieldbus ou HART.

Les normes de produit ne doivent pas nécessairement comprendre des aspects d'évaluation de conformité (comprenant des dispositions QM), normatifs ou informatifs, autres que des dispositions pour l'essai de produit (évaluation et examen).

5 Conventions pour les structures lexicales

Le langage EDDL est généralement décrit à l'aide de structures lexicales dans lesquelles les éléments et la présence de champs sont spécifiés. Une forme générale des structures lexicales est présentée ci-dessous.

ABC champ1, champ2

ABC est un élément lexical. Cet élément doit être codé dans une syntaxe concrète. Il n'est pas nécessaire de coder cet élément avec le nom "ABC". Il est également possible de coder cet élément (avec un numéro de balise (Tag), par exemple).

Champ1 et champ2 sont des champs de l'élément lexical ABC. Chaque champ est obligatoire et peut avoir plus d'un attribut. Si un champ a des attributs, la présence des attributs est spécifiée dans une table. Une virgule sépare champ1 et champ2. La virgule est un élément lexical et n'est pas explicitement codée.

Si un champ contient des attributs supplémentaires, ils sont définis dans une table. La disposition de la table et les qualificatifs d'usage possibles sont présentés au Tableau 1.

Tableau 1 – Descriptions d'attribut de champ

Usage	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 d'usage). 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 d'usage). Un seul des attributs sélectionnables (z1 ou z2) est présent
c	uuu	La présence de cet attribut est conditionnelle et celui-ci est présent uniquement si la condition est vraie

² Conformément aux directives ISO/CEI.

Les caractères dans la colonne Usage ont les définitions suivantes.

- m: cet attribut est obligatoire et doit être présent.
- o: cet attribut est facultatif et peut ne pas être présent.
- s: cet attribut est une sélection. Un et un seul des champs marqués avec "s" (Z1 ou Z2) doit être présent.
- c: cet attribut est conditionnel. La condition est décrite dans la colonne Description.

Dans la colonne Attribut, s'il existe plusieurs attributs, qui ont le même usage, ceux-ci sont triés par ordre alphabétique.

ABC champ1+

Le signe plus (+) après champ1 est utilisé pour indiquer que champ1 est utilisé au moins une fois. Celui-ci peut être utilisé plusieurs fois.

ABC champ2*

L'astérisque (*) derrière champ2 est utilisé pour indiquer que champ2 est facultatif et peut ne pas être utilisé. S'il est utilisé, il peut l'être plusieurs fois.

ABC [champ1, champ2]+

Les éléments champ1 et champ2 entre les crochets [] sont une liste non triée. Le signe plus derrière le crochet final indique que champ1 et champ2 sont utilisés au moins une fois et peuvent l'être plusieurs fois en tant que groupe.

ABC champ1, (champ2, champ3)<exp>

<exp> indique que champ2 et champ3 sont utilisés conjointement avec l'expression conditionnelle (des constructions conditionnelles sont spécifiées en 7.32). L'expression <exp> désigne uniquement les champs entre parenthèses (). L'utilisation d'expressions conditionnelles est facultative.

6 Modèle EDD et EDDL

6.1 Présentation des EDD et d'EDDL

Une description de dispositif électronique (EDD) contient tous le(s) paramètre(s) d'un dispositif d'un système d'automatismes. La technologie utilisée pour décrire une EDD est appelée langage de description électronique d'appareil (EDDL). Le langage EDDL comporte un ensemble d'éléments de langage basiques évolutifs pour gérer des dispositifs simples, complexes ou modulaires. Le langage EDDL est un langage descriptif basé sur un format ASCII avec une séparation nette entre les données et le programme.

NOTE Les données d'un champ de texte, qui sont marquées avec un code de pays (japonais, par exemple) peuvent utiliser un code multi-octets.

6.2 Architecture EDD

Du point de vue du modèle ISO/OSI (ISO/CEI 7498-1), une EDD se trouve au-dessus de la Couche 7. Toutefois, l'application EDD utilise le système de communication pour transférer ses informations. Une EDD contient des constructions qui prennent en charge le mapping du système de communication pris en charge.

Le fabricant de l'appareil définit les objets, qui forment la représentation logique des objets d'une application EDD. Pour cette raison, EDDL comprend des éléments de langage, qui

mettent en correspondance les données de l'EDD avec la représentation de données du système de communication, de sorte que l'utilisateur d'une EDD n'a pas besoin de connaître l'emplacement physique (adresse) de l'objet d'appareil.

Une EDD décrit la gestion des informations à afficher à l'utilisateur. La représentation spécifique de ce type de visualisation ne fait pas partie des définitions d'EDD ou EDDL.

6.3 Concepts d'EDD

Le fabricant d'un appareil ou d'un composant d'un système d'automatismes décrit les propriétés de l'appareil en utilisant l'EDDL. L'EDD obtenu contient des informations, telles que:

- la description des paramètres de l'appareil;
- la description des dépendances des paramètres;
- le regroupement logique des paramètres de l'appareil;
- la sélection et l'exécution des fonctions de l'appareil prises en charge;
- la description des ensembles des données transférées.

Suivant l'utilisation requise, l'EDD peut être physiquement située

- dans un dispositif;
- dans un support de stockage de données externe tel qu'un disque compact, une disquette ou un serveur;
- partiellement distribuée dans l'appareil et 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 plusieurs dictionnaires pour une EDD.

Une mise en œuvre d'EDD contient les informations nécessaires relatives à l'appareil cible (le fabricant, le type d'appareil, la révision, par exemple). Cela permet de mettre en correspondance une EDD spécifique avec un appareil spécifique.

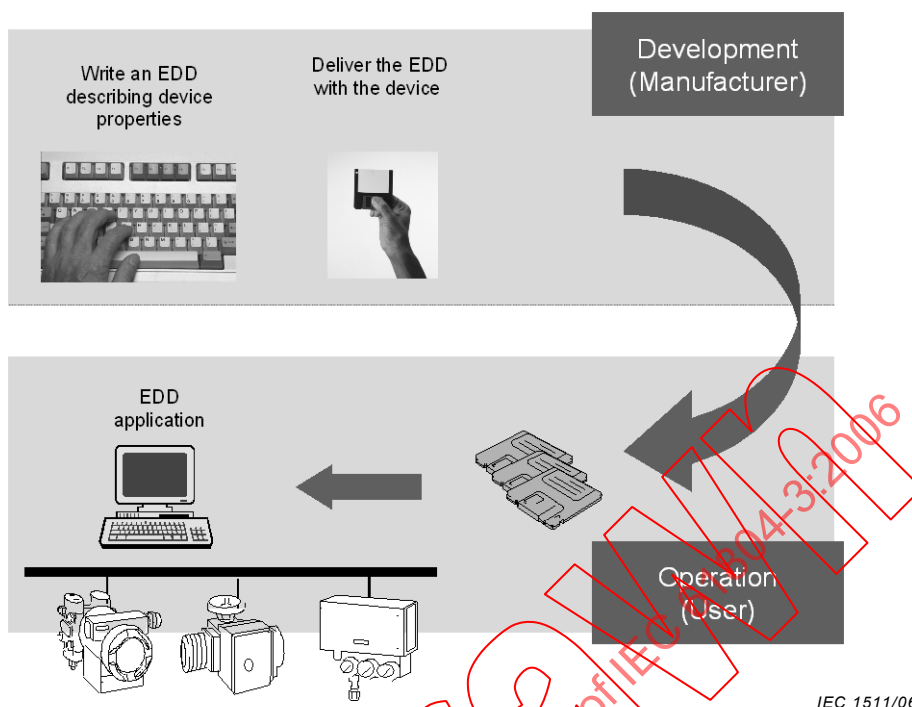
6.4 Principes du processus de développement d'EDD

6.4.1 Généralités

Le processus de création d'une EDD comporte trois étapes: la génération du code source de l'EDD, le prétraitement de l'EDD et la compilation de l'EDD.

6.4.2 Génération de source d'EDD

Le processus de génération d'une EDD est présenté à la Figure 2.



IEC 1511/06

Légende

Anglais	Français
Write an EDD describing device properties	Ecrire une EDD décrivant les propriétés de l'appareil
Deliver the EDD with the device	Livrer l'EDD avec l'appareil
Development (Manufacturer)	Développement (Fabricant)
EDD application	Application EDD
Operation (User)	Exploitation (Utilisateur)

Figure 2 – Processus de génération d'EDD

Le fabricant d'un appareil écrit une EDD appropriée pour son appareil et fournit les deux (l'EDD et l'appareil). Si le système d'automatisation prend en charge la méthode EDD, une intégration de système peut être effectuée par l'utilisateur.

Les EDD des appareils peuvent être embarquées dans la mémoire de l'appareil ou fournies à l'aide de supports de stockage séparés ou téléchargées depuis un serveur réseau approprié. L'EDD est "interprétée" ou "parcourue" par une application EDD. Les EDD sont normalement stockées en tant que fichiers sources ou fichiers prétraités.

6.4.3 Prétraitement d'EDD

Dans l'étape de prétraitement, un préprocesseur d'EDD génère une représentation de l'EDD cohérente et adaptée pour une compilation finale.

Le prétraitement prend en charge, par exemple, la substitution des définitions et l'inclusion de texte externe. La sortie du préprocesseur est une EDD complète sans aucune directive de préprocesseur.

6.4.4 Compilation d'EDD

L'étape de compilation d'EDD produit une représentation interne d'application EDD à partir d'une EDD prétraitée destinée à être utilisée dans l'application EDD.

6.5 Interrelations entre la structure lexicale et les définitions formelles

La structure lexicale de l'EDDL et ses éléments sont décrits à l'Article 7. Les définitions formelles et la syntaxe de chaque élément sont données à l'Annexe A. La structure lexicale et sa description formelle portent le même nom.

NOTE Une autre spécification peut être développée en option supplémentaire à la place des définitions formelles et de la syntaxe de l'Annexe A.

6.6 Builtins

Les Builtins sont des sous-programmes prédéfinis qui sont exécutés dans l'application EDD.

EXEMPLE Un terminal portatif est un appareil simple ayant un petit écran et des fonctions de curseur limitées. Pour ce type d'appareil, un Builtin peut être spécifié pour produire une entrée d'affichage en utilisant uniquement des actions de curseur haut/bas gauche/droite.

La bibliothèque de Builtin est définie à l'Annexe B.

6.7 Profils

EDDL est une spécification harmonisée de concepts EDD hérités existants. Les applications EDD concrètes utilisent un sous-ensemble de la spécification EDDL. Les éléments EDDL et les Builtins sont sélectionnés à partir des profils définis à l'Annexe D.

En plus des profils de l'EDD, les consortiums implémenteurs publient également des "profils d'appareils", qui sont utilisés pour prendre en charge l'interchangeabilité d'appareils conformes. Ces profils d'appareils peuvent être décrits à l'aide de l'EDDL spécifié dans le présent document.

7 Langage de description de dispositif électronique

7.1 Vue d'ensemble

7.1.1 Caractéristiques du langage EDDL

EDDL est un langage structuré et interprétatif pour décrire les propriétés d'un dispositif. De plus, les interactions entre les dispositifs et l'environnement d'exécution de l'EDD sont incorporés dans EDDL. Le langage EDDL comprend un ensemble d'éléments de langage pour ces applications.

Pour une implémentation d'EDD spécifique, il n'est pas nécessaire d'utiliser tous les éléments fournis par le langage.

Des sous-ensembles compatibles du langage EDDL sont autorisés et peuvent être spécifiés en utilisant des profils (par exemple, le choix de constructions, le nombre de récurrences, et la sélection d'options). Les profils pris en charge par certains consortiums industriels sont spécifiés à l'Annexe D. Les développeurs d'EDD doivent identifier dans chaque appareil les détails concernant le profil qui a été utilisé.

7.1.2 Représentation de syntaxe

La spécification du langage EDDL dans cet article de la norme utilise une syntaxe abstraite. La structure lexicale abstraite de cette section est convertie en syntaxes spécifiques spécifiées à l'Annexe A par le nom de l'élément (VARIABLE dans la structure lexicale équivaut au mot-clé "VARIABLE" dans l'annexe, par exemple).

NOTE Hormis la syntaxe définie dans la présente norme, il est possible d'utiliser d'autres définitions de syntaxe, qui peuvent être ajoutées dans le futur pour permettre d'autres caractéristiques et représentations pour une évolution future.

7.1.3 Eléments de langage d'une EDD

Le langage est structuré dans les éléments de langage suivants:

- élément 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 d'étiquettes, par exemple) dans un dictionnaire de texte (voir 7.36).

Les informations d'identification doivent être les premières entrées dans chaque fichier EDD et doivent apparaître une seule fois. Les informations d'identification identifient de manière unique la version de l'EDDL utilisée, ainsi que le type d'appareil, les codes du modèle et les détails de révision spécifiques couverts par le fichier EDD.

7.1.4 Eléments de construction de base

Ces constructions de base ont été spécifiées pour prendre en charge les descriptions d'appareils utilisés dans les applications de contrôle industriel conjointement avec leurs propriétés et fonctionnalités.

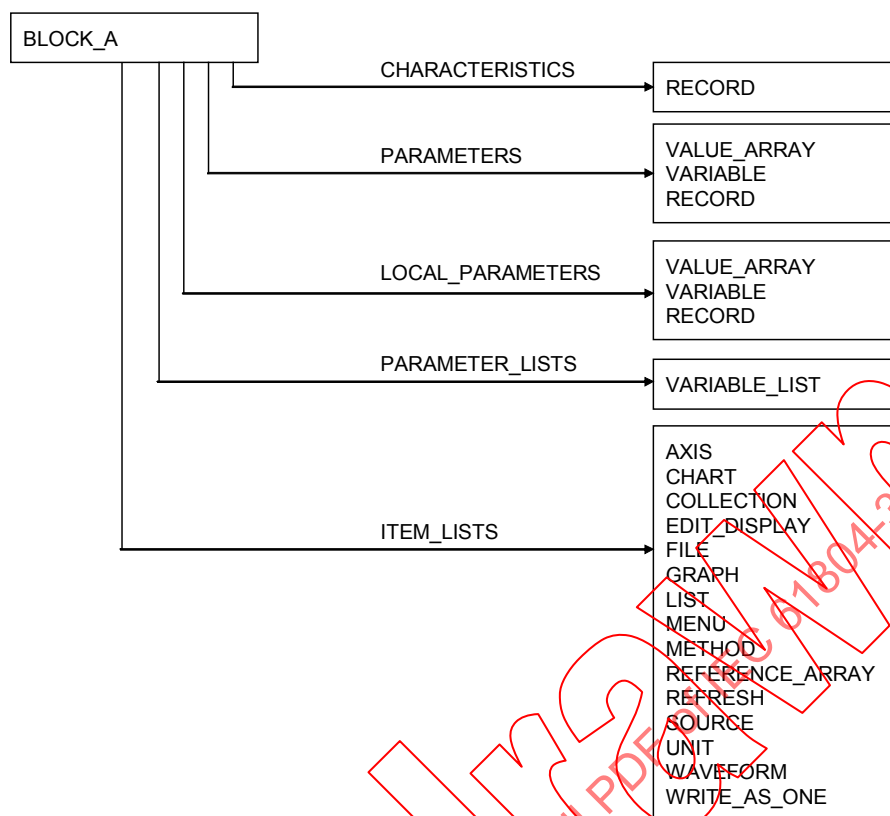
Certaines constructions ont des noms et fonctions similaires, mais qui sont différents dans leurs spécifications détaillées. Cette variété additionnelle a été incluse afin d'assurer la compatibilité avec plusieurs langages de description existants. Des références croisées de mapping appropriées sont données par les profils.

Chacune des constructions de base a un ensemble d'attributs associés à celle-ci. Les attributs peuvent avoir des sous-attributs, qui affinent la définition de l'attribut et par conséquent, la définition de la construction elle-même.

La définition d'un attribut peut être statique ou dynamique. Une définition d'attribut statique ne change jamais, tandis qu'une définition d'attribut dynamique peut changer afin de prendre en compte des changements de valeur de paramètre.

Certaines constructions de base font référence à d'autres constructions de base. Les constructions de base et leurs relations figurent dans la liste suivante:

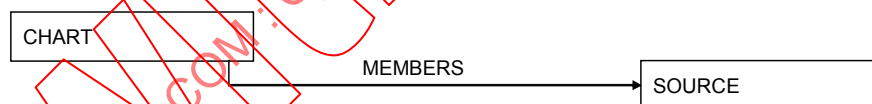
- **AXIS** décrit l'axe d'un CHART ou GRAPH (voir 7.3).
- **BLOCK_A** est un regroupement logique de CHARACTERISTICS, PARAMETERS, PARAMETER_LISTS et ITEM_LISTS (voir Figure 3). Pour accéder à un élément de BLOCK_A, il convient que l'instance du bloc soit utilisée (voir 7.4.1 et Figure 3).



IEC 1512/06

Figure 3 – BLOCK_A

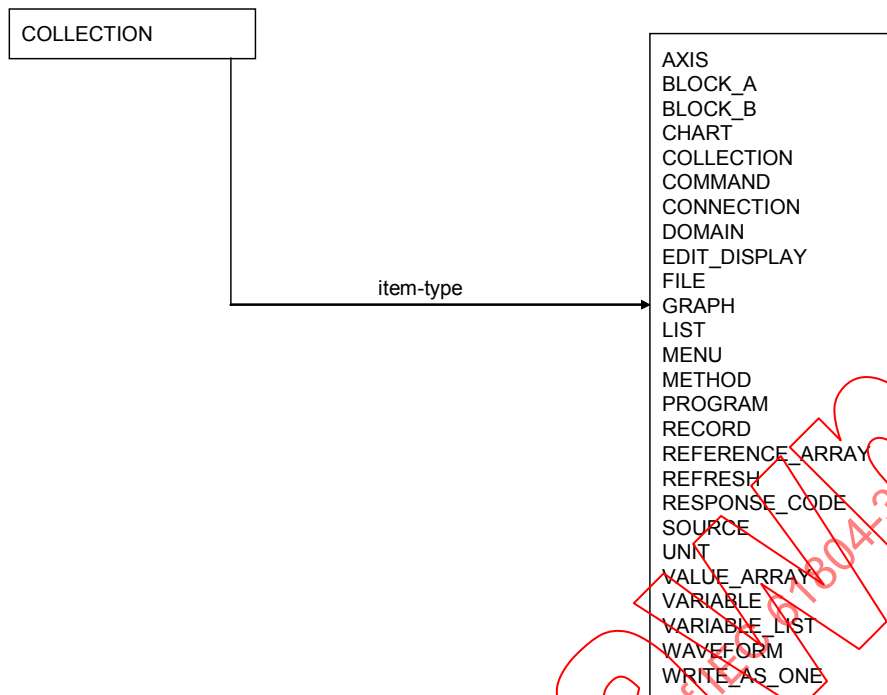
- **BLOCK_B** permet de structurer les variables en instances de types de bloc logique. Pour accéder à une **VARIABLE** dans une construction **BLOCK_B**, la construction de base **COMMAND** est utilisée pour générer l'adressage relatif (voir 7.4.2).
- **CHART** décrit un diagramme utilisé pour afficher des données provenant d'un appareil dans une application EDD (voir 7.5 et Figure 4).



IEC 1513/06

Figure 4 – CHART

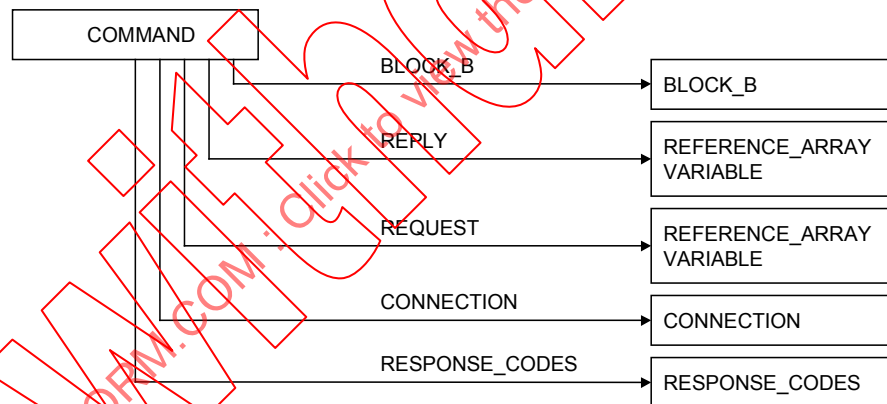
- **COLLECTION** décrit des regroupements logiques de données (voir 7.6 et Figure 5).



IEC 1514/06

Figure 5 – COLLECTION

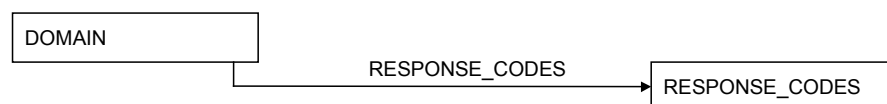
- **COMMAND** décrit la structure et l'adressage des variables dans l'appareil (voir 7.7 et Figure 6).



IEC 1515/06

Figure 6 – COMMAND

- **CONNECTION** définit plusieurs applications dans un appareil (voir 7.8).
- **DOMAIN** peut être utilisé pour télécharger ou transmettre des quantités modérément élevées de données vers ou depuis un appareil (voir 7.9 et Figure 7).



IEC 1516/06

Figure 7 – DOMAIN

- **EDIT_DISPLAY** décrit la présentation des données à un utilisateur par un appareil d'affichage (voir 7.10 et Figure 8).

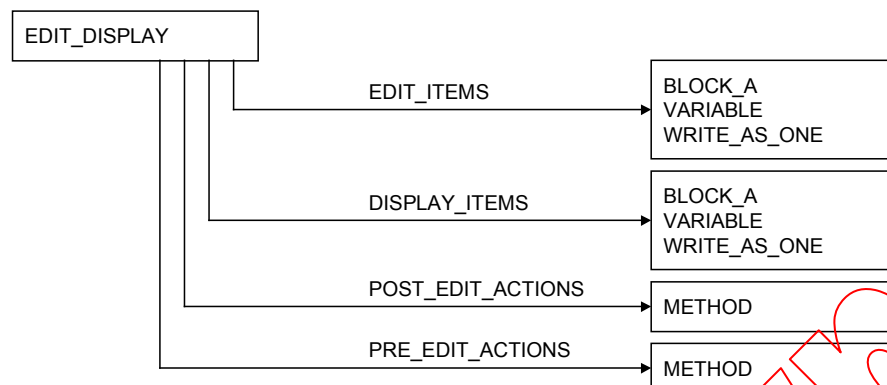


Figure 8 – EDIT_DISPLAY

IEC 1517/06

- **FILE** décrit une zone de stockage persistant (voir 7.11 et Figure 9).

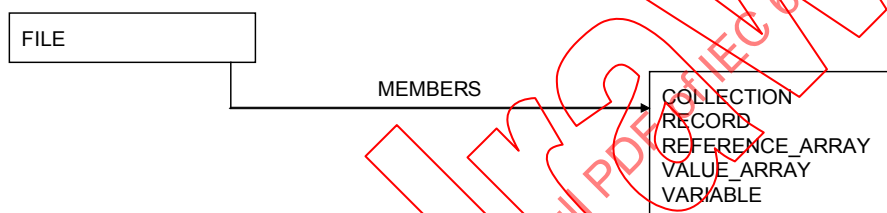


Figure 9 – FILE

IEC 1518/06

- **GRAPH** décrit un graphique utilisé pour afficher des données provenant d'un appareil dans l'application EDD (voir 7.12 et Figure 10).

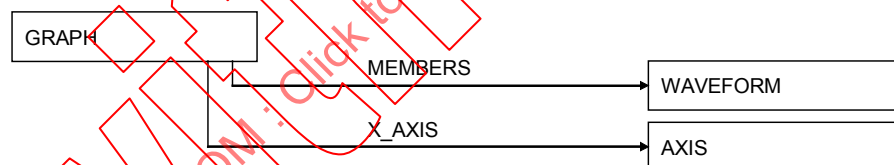
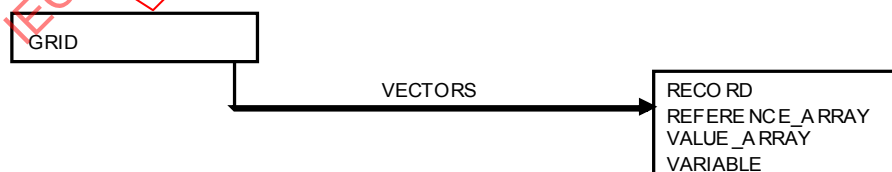


Figure 10 – GRAPH

IEC 1519/06

- **GRID** décrit une matrice utilisée pour afficher les données d'un appareil dans l'application EDD (voir 7.13 et Figure 11).



IEC 1520/06

Figure 11 – GRID

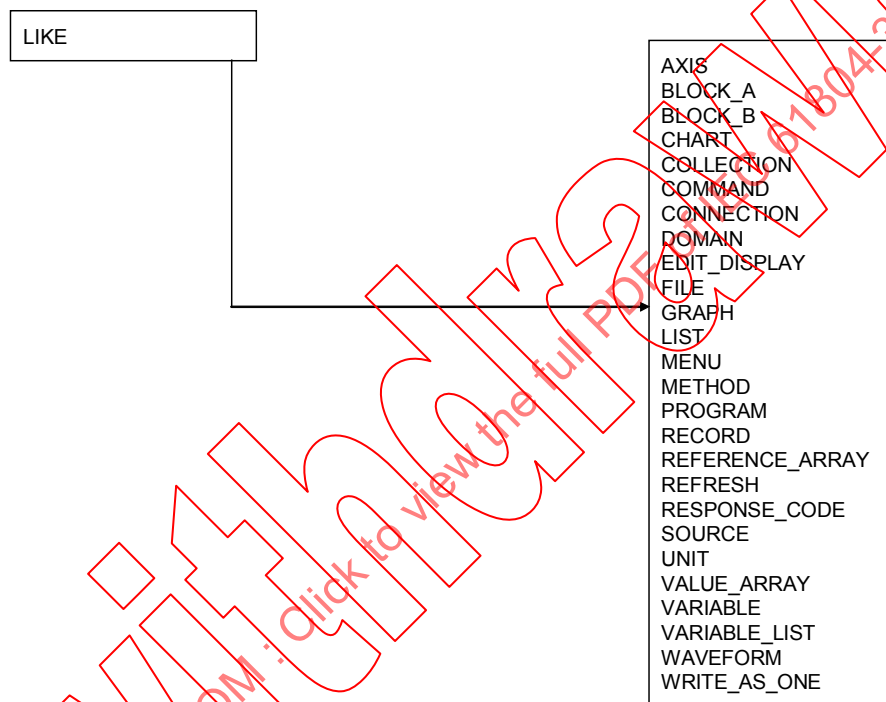
- **IMAGE** décrit une image visuelle (voir 7.13 et Figure 12).



IEC 1521/06

Figure 12 – IMAGE

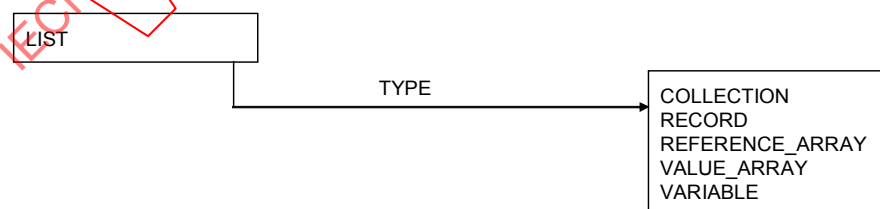
- **IMPORT** est utilisé pour importer une EDD et modifier ses définitions existantes (voir 7.13).
- **LIKE** crée une nouvelle instance d'une instance existante d'une construction de base. Les attributs de la nouvelle instance peuvent être redéfinis, ajoutés ou supprimés (voir 7.16 et Figure 13).



IEC 1522/06

Figure 13 – LIKE

- **LIST** décrit une séquence d'éléments (voir 7.17 et Figure 14).



IEC 1523/06

Figure 14 – LIST

- **MENU** décrit la présentation des données à un utilisateur par une application EDD (voir 7.18 et Figure 15).

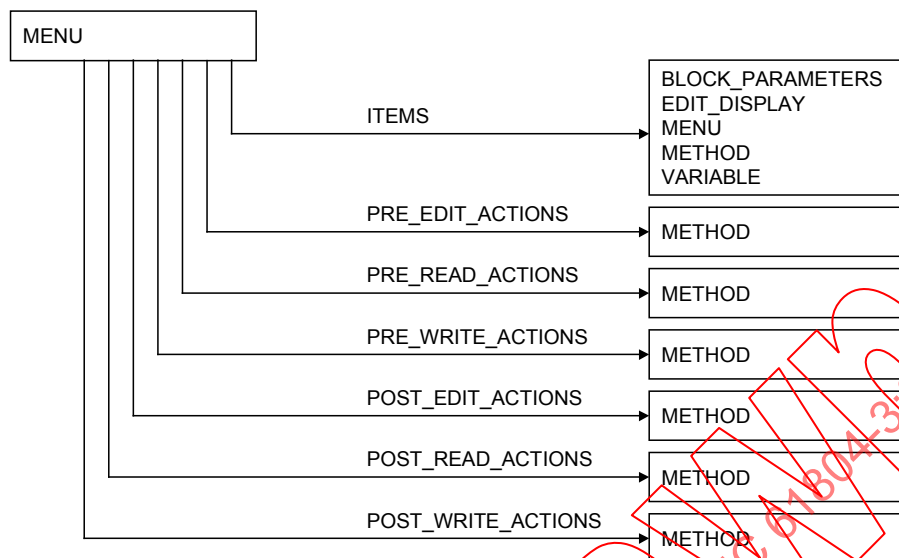
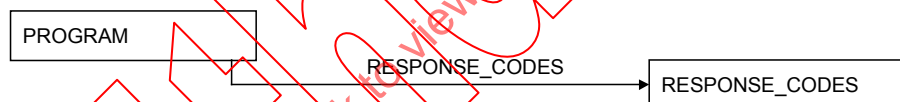


Figure 15 – MENU

IEC 1524/06

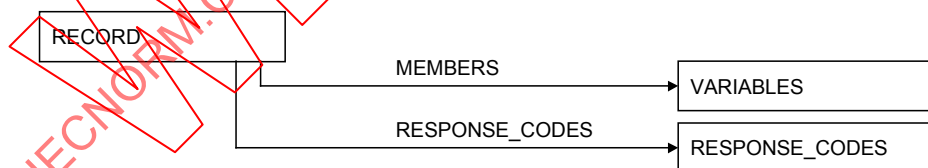
- **METHOD** décrit l'exécution d'une séquence complexe d'interactions d'événements qui doit se produire entre des appareils du système tels que des unités d'affichage, des configurateurs et des appareils de terrain (voir 7.19).
- **PROGRAM** spécifie comment un code exécutable d'appareil peut être initié par un agent approprié (voir 7.20 et Figure 16).



IEC 1525/06

Figure 16 – PROGRAM

- **RECORD** décrit les données contenues dans l'appareil (voir 7.21 et Figure 17).



IEC 1526/06

Figure 17 – RECORD

- **REFERENCE_ARRAY** est un ensemble d'éléments EDD du même type d'élément EDDL (VARIABLE ou MENU, par exemple). Un élément peut être référencé dans l'EDD via le nom de REFERENCE_ARRAY et l'index associé à l'élément (voir 7.22 et Figure 18).

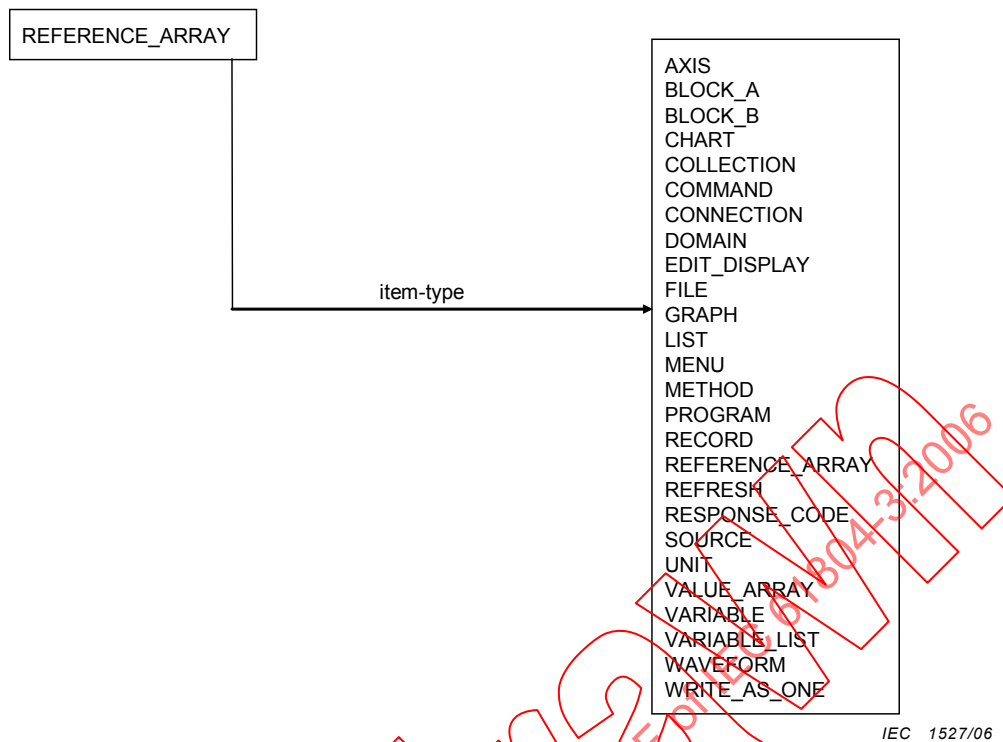


Figure 18 – REFERENCE_ARRAY

- **Relation:** trois types de relations sont spécifiés.
- REFRESH décrit les relations entre les VARIABLE, les RECORD et les VALUE_ARRAY (voir 7.23.2 et Figure 19).

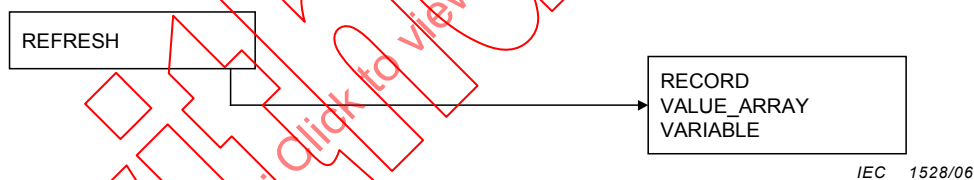


Figure 19 – REFRESH

- UNIT décrit les relations entre les VARIABLE, les RECORD et les VALUE_ARRAY (voir 7.23.3 et Figure 20).



Figure 20 – UNIT

- La relation WRITE_AS_ONE décrit les relations entre les VARIABLE, les RECORD et les VALUE_ARRAY (voir 7.23.4 et Figure 21).



Figure 21 – WRITE_AS_ONE

- **RESPONSE_CODES** spécifie les codes de réponse spécifiques à l'application pour utilisation lors de la gestion d'un VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST, PROGRAM ou DOMAIN (voir 7.24).
- **SOURCE** (voir 7.25 et Figure 22).

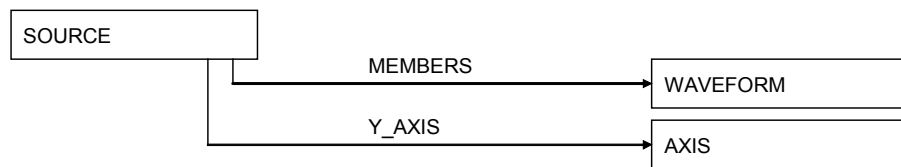


Figure 22 – SOURCE

IEC 1531/06

- **VALUE_ARRAY** décrit des groupes logiques de variables EDDL (voir 7.26 et Figure 23).

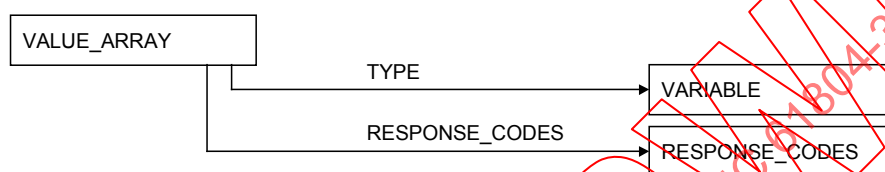


Figure 23 – VALUE_ARRAY

IEC 1532/06

- **VARIABLE** décrit les données contenues dans l'appareil (voir 7.27 et Figure 24).

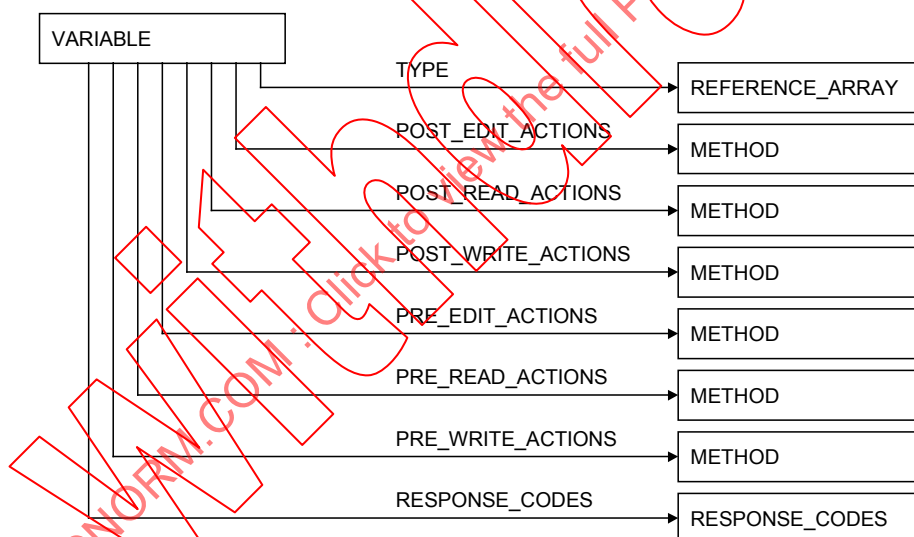


Figure 24 – VARIABLE

IEC 1533/06

- **VARIABLE_LIST** décrit les regroupements logiques de données contenues dans l'appareil qui peuvent être communiqués sous forme de liste (voir 7.28 et Figure 25).

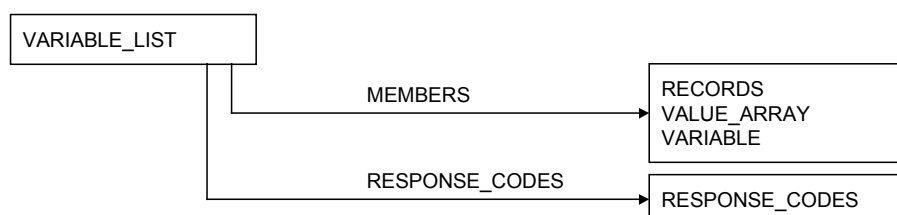
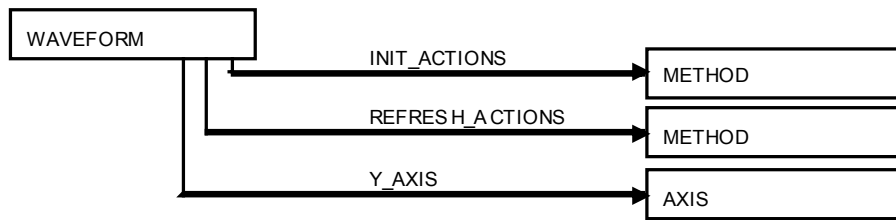


Figure 25 – VARIABLE_LIST

IEC 1534/06

- **WAVEFORM** (voir 7.29 et Figure 26).



IEC 1535/06

Figure 26 – WAVEFORM

7.1.5 Attributs communs

Les attributs communs suivants sont utilisés dans de nombreuses constructions de base:

- DEFINITION
- MEMBERS
- LABEL
- HELP
- RESPONSE_CODES

7.1.6 Éléments spéciaux

Les éléments spéciaux sont des mécanismes EDDL additionnels utilisés pour prendre en charge la gestion des fichiers, les instances multiples et la modification d'éléments de base EDDL, tels que:

- **Redirection de sortie**
 - **OPEN** permet de créer un fichier de sortie distinct pour une EDD.
 - **CLOSE** permet de fermer un fichier de sortie ouvert pour une EDD.
- Les **expressions conditionnelles** spécifient des valeurs d'attributs, qui dépendent des valeurs de variables lors de l'exécution
- La **référence** est utilisée tout au long d'une description d'appareil pour faire référence à d'autres éléments de la même EDDL
- L'**expression** est une opération logique ou mathématique sur des valeurs de variable afin de modifier les valeurs d'attribut d'une construction de base lors de l'exécution

7.1.7 Règles pour les instances

Chaque instance d'une construction de base doit être identifiée par un identifiant de type chaîne de caractères. Cet identifiant doit être unique dans l'ensemble de l'EDD. Toutes les constructions de l'EDDL représentent des instances réelles d'informations de l'appareil. Si un dispositif contient deux éléments d'information ou plus qui ont la même structure, la construction appropriée doit être répétée dans l'EDD avec des identifiants différents. En variante, la construction LIKE peut être utilisée pour créer une copie de la construction avec un identifiant différent. L'identifiant dans la partie de définition lexicale n'est pas décrit à nouveau dans chaque construction.

7.1.8 Règles pour une liste de VARIABLES

Une liste de VARIABLES 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 donnée en référence.

7.2 Informations d'identification d'une EDD

7.2.1 Structure générale

Objet

Les informations d'identification d'une EDD identifient de façon unique la description de l'appareil d'un type d'appareil spécifique d'un fabricant d'appareil. Elles spécifient également la version de l'EDDL et le profil EDDL utilisés.

Ces informations d'identification doivent constituer la première entrée dans chaque EDD et doivent apparaître une seule fois.

Structure lexicale

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

NOTE 1 Les informations d'identification d'une EDD ne sont pas structurées de la même façon que d'autres constructions EDDL.

NOTE 2 L'ordre strict des mots clés des informations d'identification d'une EDD n'est pas impératif.

7.2.2 Attributs spécifiques

7.2.2.1 DD_REVISION

Objet

Cet attribut spécifie un code unique pour la révision d'une EDD de l'appareil, tel que défini par le fabricant.

NOTE 1 Il convient que le fabricant modifie le numéro de révision chaque fois qu'une nouvelle description d'appareil est émise pour un dispositif.

NOTE 2 Il peut exister plusieurs EDD pour une révision particulière d'un appareil particulier, auquel cas l'attribut DD_REVISION permet de les distinguer.

Structure lexicale

DD_REVISION integer

L'attribut de DD_REVISION est spécifié au Tableau 2.

Tableau 2 – Attribut DD_REVISION

Usage	Attribut	Description
m	integer	Entier non signé à deux octets contenant le code pour une révision de fichier EDD spécifique

7.2.2.2 DEVICE_REVISION

Objet

Cet attribut spécifie un code unique pour la révision de l'appareil de terrain, tel que défini par le fabricant.

Structure lexicale

DEVICE_REVISION integer

L'attribut de DEVICE_REVISION est spécifié au Tableau 3.

Tableau 3 – Attribut DEVICE_REVISION

Usage	Attribut	Description
m	integer	Entier non signé à deux octets contenant le code pour une révision d'appareil spécifique

7.2.2.3 DEVICE_TYPE**Objet**

Cet attribut spécifie un identifiant pour le type d'appareil, tel que défini par le fabricant de l'appareil. Il convient que l'identifiant soit unique pour chaque type.

NOTE Le type d'appareil peut spécifier une catégorie d'appareils ou un produit spécifique.

Structure lexicale

DEVICE_TYPE (integer, identifier)

L'attribut de DEVICE_TYPE est spécifié au Tableau 4.

Tableau 4 – Attribut DEVICE_TYPE

Usage	Attribut	Description
s	integer	Entier non signé à deux octets contenant le code pour un type d'appareil ou groupe de types d'appareil spécifique
s	identifier	Référence à un entier à deux octets qui contient le code pour un type d'appareil ou groupe de types d'appareil spécifique
NOTE Le mapping de l'identifiant à une constante entière peut être stocké dans un fichier séparé appelé "devices.cfg". Ce fichier contient une liste d'identifiants pour des appareils ou groupes de types d'appareil.		

7.2.2.4 EDD_PROFILE**Objet**

Cet attribut spécifie le profil sélectionné de l'EDDL, tel que défini à l'Annexe D.

Structure lexicale

EDD_PROFILE integer

L'attribut de EDD_PROFILE est spécifié au Tableau 5.

Tableau 5 – Attribut EDD_PROFILE

Usage	Attribut	Description
o	integer	Entier non signé à deux octets contenant le code pour les profils spécifiques sélectionnés à l'Annexe D. Les codes suivants sont actuellement définis: 0x01 Profil pour HART Communication Foundation (voir Article D.4) 0x02 Profil pour Fieldbus Foundation (voir Article D.3) 0x03 Profil pour PROFIBUS (voir Article D.2)

7.2.2.5 EDD_VERSION

Objet

Ce mot-clé spécifie un code unique pour la version de l'EDD, qui a été utilisé par le fabricant dans la construction de l'EDD. Cet attribut doit être inclus avec toutes les descriptions d'appareil conformes à la présente norme.

Structure lexicale

EDD_VERSION integer

L'attribut de EDD_VERSION est spécifié au Tableau 6.

Tableau 6 – Attribut EDD_VERSION

Usage	Attribut	Description
o	integer	Entier non signé à deux octets contenant le chiffre 1 pour la première édition de la norme

7.2.2.6 MANUFACTURER

Objet

Cet attribut 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 de EDD_PROFILE et de MANUFACTURER doit être unique.

Structure lexicale

MANUFACTURER [(long integer, identifiant)<exp>]

L'attribut de MANUFACTURER est spécifié au Tableau 7.

Tableau 7 – Attribut MANUFACTURER

Usage	Attribut	Description
s	long integer	Entier long non signé contenant le code pour un fabricant spécifique Si la valeur 1 est attribuée à tous les bits de la valeur de l'entier, la structure lexicale "MANUFACTURER_EXT" doit être présente, et identifie le fabricant
sm	identifiant	Référence à un entier long non signé qui contient le code pour un fabricant spécifique
NOTE Il convient de stocker le mapping de l'identifiant avec une constante entière dans un fichier séparé appelé "devices.cfg". Ce fichier contient une liste d'identifiants pour les fabricants.		

7.2.2.7 MANUFACTURER_EXT

Objet

Cet attribut est réservé à un usage ultérieur. Il s'agit d'un espace réservé pour une chaîne permettant l'identification globale du MANUFACTURER.

NOTE La spécification de cet attribut est incluse lorsqu'elle est établie au niveau international.

Structure lexicale

MANUFACTURER_EXT string

L'attribut de MANUFACTURER_EXT est spécifié au Tableau 8.

Tableau 8 – Attribut de MANUFACTURER_EXT

Usage	Attribut	Description
c	string	Chaîne contenant les informations pour l'identification mondiale unique d'un fabricant

7.3 AXIS

7.3.1 Structure générale

Objet

AXIS (axe) décrit un axe d'un objet CHART (diagramme) ou GRAPH (graphique).

Structure lexicale

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

Les attributs d'AXIS sont spécifiés au Tableau 9.

Tableau 9 – Attributs d'AXIS

Usage	Attribut	Description
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)
o	MAX_VALUE	Élément lexical (voir 7.3.2.1)
o	MIN_VALUE	Élément lexical (voir 7.3.2.2)
o	SCALING	Élément lexical (voir 7.3.2.3)
o	CONSTANT_UNIT	Élément lexical (voir 7.3.2.4)

7.3.2 Attributs spécifiques

7.3.2.1 MAX_VALUE

Objet

L'attribut MAX_VALUE spécifie la valeur associée à la dernière marque de graduation de l'AXIS (axe). La dernière marque de graduation d'un X_AXIS et un Y_AXIS est la marque de graduation la plus à droite et la plus haute, respectivement. Par défaut, la valeur associée à la dernière marque de graduation d'un AXIS sans attribut MAX_VALUE est déduite des WAVEFORM sur le GRAPH.

Structure lexicale

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

Les attributs de MAX_VALUE sont spécifiés au Tableau 10.

Tableau 10 – Attribut de MAX_VALUE, MIN_VALUE

Usage	Attribut	Description
o	reference	Référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY ou un élément d'une instance de REFERENCE_ARRAY
o	double	Constante à virgule flottante à double précision
o	float	Constante à virgule flottante à simple précision
o	integer	Constante entière
o	unsigned_integer	Constante entière non signée

7.3.2.2 MIN_VALUE

Objet

L'attribut MIN_VALUE spécifie la valeur associée à la première marque de graduation de l'AXIS. La première marque de graduation d'un X_AXIS et un Y_AXIS est la marque de graduation la plus à gauche et la plus basse, respectivement. Par défaut, la valeur associée à la première marque de graduation d'un AXIS sans attribut MIN_VALUE est déduite des WAVEFORM sur le GRAPH.

Structure lexicale

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

Les attributs de MIN_VALUE sont spécifiés au Tableau 10.

7.3.2.3 SCALING

Objet

L'attribut SCALING (mise à l'échelle) spécifie la façon selon laquelle il convient que les marques de graduation de l'AXIS soient mises à l'échelle. Par défaut, un AXIS sans attribut SCALING est mis à l'échelle de façon linéaire.

Structure lexicale

SCALING (LINEAR, LOGARITHMIC)<exp>

Les attributs de SCALING sont spécifiés au Tableau 11.

Tableau 11 – Attribut de SCALING

Usage	Attribut	Description
s	LINEAR	Spécifie qu'il convient de mettre à l'échelle de façon linéaire les marques de graduation sur l'axe
s	LOGARITHMIC	Spécifie qu'il convient de mettre à l'échelle de façon logarithmique (base 10) les marques de graduation sur l'axe

7.3.2.4 CONSTANT_UNIT

Objet

L'attribut CONSTANT_UNIT est utilisé, si un AXIS a un code unitaire qui ne change jamais. CONSTANT_UNIT est spécifiée comme étant une chaîne qui est affichée avec la valeur. Un AXIS sans unité constante n'a pas d'unités associées ou bien les unités ne sont pas constantes.

Structure lexicale

CONSTANT_UNIT (string)<exp>

L'attribut de CONSTANT_UNIT est spécifié au Tableau 150.

7.4 BLOCK

7.4.1 BLOCK_A

7.4.1.1 Structure générale

Objet

Un dispositif peut être organisé en blocs logiques. Chaque BLOCK_A est un regroupement logique de PARAMETERS (paramètres) et d'informations additionnelles pour des applications

BLOCK_A. Les types de blocs sont décrits par l'attribut CHARACTERISTICS. Plusieurs BLOCK_A du même type peuvent être instanciés.

NOTE Cette approche du BLOCK_A est optimisée pour l'efficacité d'accès.

Structure lexicale

BLOCK_A identifieur [CHARACTERISTICS, LABEL, PARAMETERS, AXIS_ITEMS, CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISPLAY_ITEMS, FILE_ITEMS, GRAPH_ITEMS, GRID_ITEMS, IMAGE_ITEMS, LIST_ITEMS, HELP, MENU_ITEMS, METHOD_ITEMS, PARAMETER_LISTS, REFERENCE_ARRAY_ITEMS, REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS, WRITE_AS_ONE_ITEMS]

Les attributs de BLOCK_A sont spécifiés au Tableau 12.

Tableau 12 – Attributs de BLOCK_A

Usage	Attribut	Description
m	CHARACTERISTICS	Élément lexical (voir 7.4.1.2.1)
m	LABEL	Élément lexical (voir 7.30.3)
m	PARAMETERS	Élément lexical (voir 7.4.1.2.2)
o	AXIS_ITEMS	Élément lexical (voir 7.4.1.2.3)
o	CHART_ITEMS	Élément lexical (voir 7.4.1.2.4)
o	COLLECTION_ITEMS	Élément lexical (voir 7.4.1.2.5)
o	EDIT_DISPLAY_ITEMS	Élément lexical (voir 7.4.1.2.6)
o	FILE_ITEMS	Élément lexical (voir 7.4.1.2.7)
o	GRAPH_ITEMS	Élément lexical (voir 7.4.1.2.8)
o	GRID_ITEMS	Éléments lexicaux (voir 7.4.1.2.9)
o	IMAGE_ITEMS	Éléments lexicaux (voir 7.4.1.2.10)
o	LIST_ITEMS	Élément lexical (voir 7.4.1.2.11)
o	LOCAL_PARAMETERS	Élément lexical (voir 7.4.1.2.12)
o	HELP	Élément lexical (voir 7.30.2)
o	MENU_ITEMS	Élément lexical (voir 7.4.1.2.13)
o	METHOD_ITEMS	Élément lexical (voir 7.4.1.2.14)
o	PARAMETER_LISTS	Élément lexical (voir 7.4.1.2.15)
o	REFERENCE_ARRAY_ITEMS	Élément lexical (voir 7.4.1.2.16)
o	REFRESH_ITEMS	Élément lexical (voir 7.4.1.2.17)
o	SOURCE_ITEMS	Élément lexical (voir 7.4.1.2.18)
o	UNIT_ITEMS	Élément lexical (voir 7.4.1.2.19)
o	WAVEFORM_ITEMS	Élément lexical (voir 7.4.1.2.20)
o	WRITE_AS_ONE_ITEMS	Élément lexical (voir 7.4.1.2.21)

7.4.1.2 Attributs spécifiques

7.4.1.2.1 CHARACTERISTICS

Objet

L'attribut CHARACTERISTICS spécifie les caractéristiques externes d'un BLOCK_A. CHARACTERISTICS peut contenir une classe, un type de fonction, un type de bloc et une heure d'exécution. Les caractéristiques externes d'un BLOCK_A sont contenues dans un RECORD.

NOTE En général, les CHARACTERISTICS de BLOCK_A fixent le type du BLOCK_A.

Structure lexicale

CHARACTERISTICS reference

L'attribut de CHARACTERISTICS est spécifié au Tableau 13.

Tableau 13 – Attribut de CHARACTERISTIC

Usage	Attribut	Description
m	reference	Référence à une instance de RECORD

7.4.1.2.2 PARAMETERS

Objet

L'attribut PARAMETERS (paramètres) contient une liste de références à des instances de VARIABLE, VALUE_ARRAY (matrice de valeurs) ou RECORD (enregistrement). Chaque référence doit avoir un identifiant et peut avoir une description et/ou un texte d'aide.

NOTE Les PARAMETERS dans un BLOCK_A correspondent à des objets de communication individuels, répertoriés dans un dictionnaire d'objets de l'appareil. Les éléments PARAMETERS font référence aux définitions de communication de ces objets. Le dictionnaire d'objets mappe dans l'emplacement spécifique des données d'un appareil.

Structure lexicale

PARAMETERS [identifiant, reference, description, help] +

Les attributs de PARAMETERS sont spécifiés au Tableau 14.

Tableau 14 – Attributs de PARAMETER

Usage	Attribut	Description
m	identifiant	Identifiant de VARIABLE, VALUE_ARRAY ou RECORD. Chaque élément de la liste PARAMETERS doit comporter un identifiant à utiliser dans l'EDD pour y faire référence
m	reference	Référence à une instance de VARIABLE, VALUE_ARRAY ou RECORD
o	description	Brève description de la référence
o	help	Spécifie le texte d'aide pour la référence

7.4.1.2.3 AXIS_ITEMS

Objet

AXIS_ITEMS est une liste des instances d'AXIS associées à un BLOCK_A.

Structure lexicale

AXIS_ITEMS reference +

L'attribut d'AXIS_ITEMS est spécifié au Tableau 15.

Tableau 15 – Attribut d'AXIS_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance d'AXIS

7.4.1.2.4 CHART_ITEMS

Objet

CHART_ITEMS est une liste des instances de CHART associées à BLOCK_A.

Structure lexicale

CHART_ITEMS reference+

L'attribut de CHART_ITEMS est spécifié au Tableau 16.

Tableau 16 – Attribut de CHART_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de CHART

7.4.1.2.5 COLLECTION_ITEMS

Objet

COLLECTION_ITEMS est une liste des instances de COLLECTION associées à un BLOCK_A.

Structure lexicale

COLLECTION_ITEMS reference+

L'attribut de COLLECTION_ITEMS est spécifié au Tableau 17.

Tableau 17 – Attribut de COLLECTION_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de COLLECTION

7.4.1.2.6 EDIT_DISPLAY_ITEMS

Objet

EDIT_DISPLAY_ITEMS est une liste des instances d'EDIT_DISPLAY associées à BLOCK_A.

Structure lexicale

EDIT_DISPLAY_ITEMS reference+

L'attribut d'EDIT_DISPLAY_ITEMS est spécifié au Tableau 18.

Tableau 18 – Attribut d'EDIT_DISPLAY_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance d'EDIT_DISPLAY

7.4.1.2.7 FILE_ITEMS

Objet

FILE_ITEMS est une liste des instances de FILE associées à un BLOCK_A.

Structure lexicale

FILE_ITEMS reference+

L'attribut de FILE_ITEMS est spécifié au Tableau 19.

Tableau 19 – Attribut de FILE_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de FILE

7.4.1.2.8 GRAPH_ITEMS

Objet

GRAPH_ITEMS est une liste des instances de GRAPH associées à un BLOCK_A.

Structure lexicale

GRAPH_ITEMS reference+

L'attribut de GRAPH_ITEMS est spécifié au Tableau 20.

Tableau 20 – Attribut de GRAPH_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de GRAPH

7.4.1.2.9 GRID_ITEMS

Objet

GRID_ITEMS est une liste des instances de GRID associées à un BLOCK_A.

Structure lexicale

GRID_ITEMS reference+

L'attribut de GRID_ITEMS est spécifié dans le Tableau 21.

Tableau 21 – Attributs de GRID_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de GRID

7.4.1.2.10 IMAGE_ITEMS

Objet

IMAGE_ITEMS est une liste des instances d'IMAGE associées à BLOCK_A.

Structure lexicale

IMAGE_ITEMS reference+

L'attribut d'IMAGE_ITEMS est spécifié dans le Tableau 22.

Tableau 22 – Attribut d'IMAGE_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance d'IMAGE

7.4.1.2.11 LIST_ITEMS

Objet

LIST_ITEMS est une liste des instances de LIST associées à BLOCK_A.

Structure lexicale

LIST_ITEMS reference+

L'attribut de LIST_ITEMS est spécifié au Tableau 23.

Tableau 23 – Attribut de LIST_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de LIST

7.4.1.2.12 LOCAL_PARAMETERS

Objet

L'attribut LOCAL_PARAMETERS (paramètres locaux) contient une liste de références à des instances de VARIABLE, VALUE_ARRAY ou RECORD. Chaque référence doit avoir un identifiant et peut avoir une description et/ou un texte d'aide.

NOTE Les LOCAL_PARAMETERS dans un BLOCK_A ne correspondent pas à des objets de communication. Ce sont des paramètres utilisés uniquement par l'application EDD.

Structure lexicale

LOCAL_PARAMETERS [identifier, reference, description, help] +

Les attributs de LOCAL_PARAMETERS sont spécifiés au Tableau 14.

7.4.1.2.13 MENU_ITEMS

Objet

La liste MENU_ITEMS spécifie les éléments MENU associés à BLOCK_A.

Structure lexicale

MENU_ITEMS reference+

L'attribut de MENU_ITEMS est spécifié au Tableau 24.

Tableau 24 – Attribut de MENU_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de MENU

7.4.1.2.14 METHOD_ITEMS

Objet

La liste METHOD_ITEMS spécifie les METHOD associés à un BLOCK_A.

Structure lexicale

METHOD_ITEMS reference+

L'attribut de METHOD_ITEMS est spécifié au Tableau 25.

Tableau 25 – Attribut de METHOD_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de METHOD

7.4.1.2.15 PARAMETER_LISTS**Objet**

PARAMETER_LISTS spécifie les VARIABLE_LIST associés à un BLOCK_A, ainsi qu'une description optionnelle et une aide pour chaque VARIABLE_LIST. Chaque BLOCK_A peut contenir plusieurs PARAMETER_LISTS.

Structure lexicale

PARAMETER_LISTS [identifiant, reference, description, help] +

Les attributs de PARAMETER_LISTS sont spécifiés au Tableau 26.

Tableau 26 – Attributs de PARAMETER_LISTS

Usage	Attribut	Description
m	identifiant	Identifiant de la VARIABLE_LIST. Chaque élément de la liste PARAMETER_LIST doit avoir un identifiant devant être utilisé dans l'EDD pour y faire référence.
m	reference	Référence à une instance de VARIABLE_LIST
o	description	Brève description de l'élément
o	help	Spécifie le texte d'aide pour l'élément

7.4.1.2.16 REFERENCE_ARRAY_ITEMS**Objet**

La liste REFERENCE_ARRAY_ITEMS spécifie les REFERENCE_ARRAY associés à un BLOCK_A.

Structure lexicale

REFERENCE_ARRAY_ITEMS reference +

L'attribut de REFERENCE_ARRAY_ITEMS est spécifié au Tableau 27.

Tableau 27 – Attribut de REFERENCE_ARRAY_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de REFERENCE_ARRAY

7.4.1.2.17 REFRESH_ITEMS**Objet**

La liste REFRESH_ITEMS spécifie les relations REFRESH associées à un BLOCK_A.

Structure lexicale

REFRESH_ITEMS reference +

L'attribut de REFRESH_ITEMS est spécifié au Tableau 28.

Tableau 28 – Attribut de REFRESH_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de REFRESH

7.4.1.2.18 SOURCE_ITEMS**Objet**

La liste SOURCE_ITEMS spécifie les éléments SOURCE associés à un BLOCK_A.

Structure lexicale

SOURCE_ITEMS reference+

L'attribut de SOURCE_ITEMS est spécifié au Tableau 29.

Tableau 29 – Attribut de SOURCE_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de SOURCE

7.4.1.2.19 UNIT_ITEMS**Objet**

La liste UNIT_ITEMS spécifie les relations des UNIT associés à un BLOCK_A.

Structure lexicale

UNIT_ITEMS reference+

L'attribut de UNIT_ITEMS est spécifié au Tableau 30.

Tableau 30 – Attribut de UNIT_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance d'UNIT

7.4.1.2.20 WAVEFORM_ITEMS**Objet**

La liste WAVEFORM_ITEMS spécifie les instances de WAVEFORM associées à un BLOCK_A.

Structure lexicale

WAVEFORM_ITEMS reference+

L'attribut de WAVEFORM_ITEMS est spécifié au Tableau 31.

Tableau 31 – Attribut de WAVEFORM_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de WAVEFORM

7.4.1.2.21 WRITE_AS_ONE_ITEMS

Objet

La liste WRITE_AS_ONE_ITEMS spécifie les relations WRITE_AS_ONE associées à un BLOCK_A.

Structure lexicale

WRITE_AS_ONE_ITEMS *reference*+

L'attribut de WRITE_AS_ONE_ITEMS est spécifié au Tableau 32.

Tableau 32 – Attribut de WRITE_AS_ONE_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de WRITE_AS_ONE

7.4.2 BLOCK_B

7.4.2.1 Structure générale

Objet

Les variables d'un dispositif ou module sont respectivement structurées en blocs correspondant aux composants ou aux parties fonctionnelles d'un dispositif. Trois types de blocs sont définis: PHYSICAL (physique), TRANSDUCER (transducteur) et FUNCTION (fonction). Plusieurs BLOCK_B du même TYPE peuvent être instanciés. Afin de faciliter l'accès, chaque instance a son propre NUMBER (numéro). Pour chacun des types de bloc, il existe différentes chaînes de séquences de NUMBER, commençant par 1.

Lors de l'accès à un VARIABLE dans un dispositif, la construction de BLOCK_B est utilisée conjointement avec COMMAND (commande) pour permettre un adressage relatif.

NOTE 1 Cette approche de BLOCK_B est optimisée pour une meilleure efficacité de la mémoire.

NOTE 2 Le stockage d'une variable dans un appareil est spécifique au fabricant et est représenté par un répertoire d'appareils. Un répertoire de dispositifs contient un résumé des entrées de BLOCK_B disponibles dans un dispositif. De plus, le répertoire de dispositifs contient pour chaque 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'appareils.

Structure lexicale

BLOCK_B *identifier* [NUMBER, TYPE]

Les attributs de BLOCK_B sont spécifiés au Tableau 33.

Tableau 33 – Attributs de BLOCK_B

Usage	Attribut	Description
m	NUMBER	Élément lexical (voir 7.4.2.2.1)
m	TYPE	Élément lexical (voir 7.4.2.2.2)

7.4.2.2 Attributs spécifiques

7.4.2.2.1 NUMBER

Objet

Un appareil peut contenir plusieurs instances de BLOCK_B, qui sont identifiées avec l'attribut NUMBER. L'attribut NUMBER (numéro) énumère chaque BLOCK_B du même TYPE dans l'appareil.

Structure lexicale

NUMBER (integer, expression)<exp>

Les attributs de NUMBER sont spécifiés au Tableau 34.

Tableau 34 – Attributs de NUMBER

Usage	Attribut	Description
s	integer	Numéro de l'instance de BLOCK_B du même TYPE
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut NUMBER (pour la description des expressions, voir 7.34.7)

7.4.2.2.2 TYPE

Objet

L'attribut TYPE est utilisé pour spécifier un des trois types de bloc.

Structure lexicale

TYPE [PHYSICAL, TRANSDUCER, FUNCTION]

Les attributs de TYPE sont spécifiés au Tableau 35.

Tableau 35 – Attributs de TYPE

Usage	Attribut	Description
s	FUNCTION	Bloc nommé composé d'un ou de plusieurs paramètres d'entrée, de sortie ou internes. Les fonctions blocs représentent les fonctions d'automatisation de base exécutées par une application EDD, qui est indépendante des dispositifs spécifiques et du réseau. Chaque bloc de fonction traite des paramètres d'entrée conformément à un algorithme spécifié et un ensemble de paramètres internes. Ils génèrent des paramètres de sortie qui sont disponibles pour utilisation dans la même application EDD de bloc de fonction ou par d'autres applications de bloc de fonction
s	PHYSICAL	Les caractéristiques spécifiques du matériel/ressource d'un appareil de terrain sont rendues visibles par l'intermédiaire du bloc physique. De manière similaire aux blocs transducteur, elles isolent des blocs de fonctions du matériel physique en contenant un ensemble de paramètres du matériel/ressource indépendants de la mise en œuvre
s	TRANSDUCER	Les blocs transducteurs isolent les blocs de fonction des éléments spécifiques de l'appareil E/S, tels que des capteurs, des actionneurs et des commutateurs. Les blocs transducteurs contrôlent l'accès à des dispositifs E/S par l'intermédiaire d'une interface indépendante des dispositifs définie pour une utilisation par des fonctions blocs. Les blocs transducteurs remplissent des fonctions (étalonnage et linéarisation, par exemple) sur des données E/S pour les convertir en représentation indépendante de l'appareil

7.5 CHART

7.5.1 Structure générale

Objet

CHART décrit un diagramme utilisé pour afficher des données provenant d'un dispositif dans l'application EDD. Un objet CHART est utilisé pour afficher des valeurs de données continues provenant de l'appareil, par opposition à un GRAPH (graphique) qui est utilisé pour afficher un ensemble de données qui est stocké dans un dispositif. Les données provenant de l'appareil peuvent être ajustées avant d'être affichées via l'utilisation d'INIT_ACTIONS et de REFRESH_ACTIONS sur les membres SOURCE du CHART.

Structure lexicale

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

Les attributs de CHART sont spécifiés au Tableau 36.

Tableau 36 – Attributs de CHART

Usage	Attribut	Description
M	MEMBERS	Élément lexical (voir 7.30.4)
o	CYCLE_TIME	Élément lexical (voir 7.5.2.1)
o	HEIGHT	Élément lexical (voir 7.5.2.2)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)
o	LENGTH	Élément lexical (voir 7.5.2.4)
o	TYPE	Élément lexical (voir 7.5.2.5)
o	WIDTH	Élément lexical (voir 7.5.2.3)

7.5.2 Attributs spécifiques

7.5.2.1 CYCLE_TIME

Objet

L'attribut CYCLE_TIME (durée de cycle) spécifie le temps, en ms, durant lequel l'application EDD doit attendre entre des lectures successives des valeurs sources de l'appareil. Par défaut, la durée de cycle d'un CHART sans attribut CYCLE_TIME est de 1 000 ms.

Structure lexicale

CYCLE_TIME (integer)<exp>

L'attribut de CYCLE_TIME est spécifié au Tableau 37.

Tableau 37 – Attribut de CYCLE_TIME

Usage	Attribut	Description
m	integer	Temps, en ms, entre des lectures successives des valeurs sources de l'appareil

7.5.2.2 HEIGHT

Objet

L'attribut HEIGHT spécifie la largeur de la taille verticale qu'il convient que CHART ait lors de son affichage. Par défaut, la taille d'un CHART sans attribut HEIGHT est MEDIUM.

Structure lexicale

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

Les attributs de HEIGHT sont spécifiés au Tableau 38.

Tableau 38 – Attributs de HEIGHT/WIDTH

Usage	Attribut	Description
s	XX_SMALL	Spécifie qu'il convient que CHART occupe une partie extrêmement petite de la fenêtre dans laquelle il est affiché.
s	X_SMALL	Spécifie qu'il convient que CHART occupe une très petite partie de la fenêtre dans laquelle il est affiché.
s	SMALL	Spécifie qu'il convient que CHART occupe une petite partie de la fenêtre dans laquelle il est affiché.
s	MEDIUM	Spécifie qu'il convient que CHART occupe une partie moyenne de la fenêtre dans laquelle il est affiché.
s	LARGE	Spécifie qu'il convient que CHART occupe une grande partie de la fenêtre dans laquelle il est affiché.
s	X_LARGE	Spécifie qu'il convient que CHART occupe une très grande partie de la fenêtre dans laquelle il est affiché.
s	XX_LARGE	Spécifie qu'il convient que CHART occupe une partie extrêmement grande de la fenêtre dans laquelle il est affiché.

7.5.2.3 WIDTH

Objet

L'attribut WIDTH spécifie la largeur de la taille horizontale qu'il convient que CHART ait lors de son affichage. Par défaut, la taille d'un CHART sans attribut HEIGHT est MEDIUM.

Structure lexicale

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

Les attributs de WIDTH sont spécifiés au Tableau 38.

7.5.2.4 LENGTH

Objet

L'attribut LENGTH (durée) spécifie la durée, en ms, qui est affichée par le CHART. Le nombre d'échantillons affichés par un diagramme peut être calculé en divisant l'attribut LENGTH par l'attribut CYCLE_TIME (durée de cycle). Par défaut, la durée de cycle d'un CHART sans attribut LENGTH est de 600 000 ms.

Structure lexicale

LENGTH (integer) <exp>

L'attribut de LENGTH est spécifié au Tableau 39.

Tableau 39 – Attribut de LENGTH

Usage	Attribut	Description
m	integer	Spécifie la durée, en ms, qui est affichée par le CHART

7.5.2.5 TYPE

Objet

L'attribut TYPE spécifie le type de CHART que l'EDD doit afficher. Par défaut, le type d'un CHART sans attribut TYPE est STRIP.

Structure lexicale

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

Les attributs de TYPE sont spécifiés au Tableau 40.

Tableau 40 – Attribut de TYPE

Usage	Attribut	Description
s	GAUGE	Une valeur source unique s'affiche sous la forme d'une jauge, qui est une représentation graphique des données similaire à la jauge de carburant d'une automobile.
s	HORIZONTAL_BAR	Les valeurs sources s'affichent sous la forme d'un diagramme à barres horizontal.
s	SCOPE	Lorsque les valeurs sources atteignent la fin de la zone d'affichage du CHART, cette dernière est effacée et les nouvelles valeurs sources sont à nouveau affichées à partir du début de la zone d'affichage.
s	STRIP	Lorsque les valeurs sources atteignent la fin de la zone d'affichage du CHART, cette dernière est décalée, les valeurs sources les plus anciennes étant effacées, et les nouvelles valeurs sources sont ajoutées à la zone d'affichage.
s	SWEEP	Lorsque les valeurs sources atteignent la fin de la zone d'affichage du CHART, les nouvelles valeurs sources s'affichent de nouveau 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.
s	VERTICAL_BAR	Les valeurs sources s'affichent sous la forme d'un diagramme à barres vertical.

7.6 COLLECTION

7.6.1 Structure générale

Objet

Une COLLECTION est un groupe logique d'éléments, tels que des VARIABLE ou un MENU. Un identifiant est assigné à chaque élément. Les éléments peuvent être référencés dans la description de l'appareil à l'aide de l'identifiant de COLLECTION et du nom de l'élément.

NOTE 1 COLLECTION est destiné à être utilisé par un outil (pour identifier des paramètres qu'il convient de traiter conjointement, par exemple).

NOTE 2 Le niveau d'imbrication pris en charge recommandé est de deux (COLLECTION de COLLECTION, par exemple).

Structure lexicale

COLLECTION identifieur, item-type, [MEMBERS, HELP, LABEL]

Les attributs de COLLECTION sont spécifiés au Tableau 41.

Tableau 41 – Attributs de COLLECTION

Usage	Attribut	Description
o	item-type	Élément lexical (voir 7.6.2.1)
m	MEMBERS	Élément lexical (voir 7.30.4)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)

7.6.2 Attributs spécifiques

7.6.2.1 item-type

Objet

L'attribut item-type spécifie le type de membres dans la collection. Si item-type est spécifié, tous les membres de collection doivent être du type spécifié. Sinon, les membres de collection peuvent être d'un type quelconque.

Structure lexicale

item-type

Le Tableau 42 présente les types d'option autorisés:

Tableau 42 – item-type

Usage	item-type	Description
s	AXIS	Sélection de cette construction de base
s	BLOCK_A	Sélection de cette construction de base
s	BLOCK_B	Sélection de cette construction de base
s	CHART	Sélection de cette construction de base
s	COLLECTION	Sélection de cette construction de base
s	COMMAND	Sélection de cette construction de base
s	CONNECTION	Sélection de cette construction de base
s	DOMAIN	Sélection de cette construction de base
s	EDIT_DISPLAY	Sélection de cette construction de base
s	FILE	Sélection de cette construction de base
s	GRAPH	Sélection de cette construction de base
s	LIST	Sélection de cette construction de base
s	MENU	Sélection de cette construction de base
s	METHOD	Sélection de cette construction de base
s	PROGRAM	Sélection de cette construction de base
s	RECORD	Sélection de cette construction de base
s	REFERENCE_ARRAY	Sélection de cette construction de base
s	REFRESH	Sélection de cette construction de base
s	RESPONSE_CODES	Sélection de cette construction de base
s	SOURCE	Sélection de cette construction de base
s	UNIT	Sélection de cette construction de base
s	VALUE_ARRAY	Sélection de cette construction de base
s	VARIABLE	Sélection de cette construction de base
s	VARIABLE_LIST	Sélection de cette construction de base

Usage	item-type	Description
s	WAVEFORM	Sélection de cette construction de base
s	WRITE_AS_ONE	Sélection de cette construction de base

7.6.2.2 Généralités

COLLECTION ne comporte qu'un seul attribut spécifique.

7.7 COMMAND

7.7.1 Structure générale

Objet

Un objet COMMAND (commande) est une construction utilisée pour prendre en charge le mapping des éléments de communication EDD avec un système de communication choisi. Il spécifie différents éléments requis pour construire une trame de communication. Si cette construction est utilisée par un profil EDDL, chaque élément de données à transmettre doit être référencé à l'intérieur d'une COMMAND et également dans le MENU de transmission/téléchargement (voir 7.18.2.10).

NOTE 1 Le champ d'adresse d'une trame de communication est spécifié à l'aide des constructions sélectionnables BLOCK, SLOT, NUMBER, avec INDEX et MODULE, selon le cas. S'il existe des instances de processus d'application EDD multiples, ou une instance de processus d'application EDD qui nécessite une référence explicite, dans un appareil, celles-ci sont spécifiées par la construction additionnelle CONNECTION (connexion). Le type de trame de communication ou son champ de contrôle est spécifié par la construction OPERATION (opération). Le contenu de données de la trame de communication est spécifié par la construction TRANSACTION.

NOTE 2 Le domaine d'application de BLOCK, SLOT, NUMBER est limité à une instance de processus d'application EDD donnée, c'est-à-dire que la même valeur de BLOCK, SLOT, NUMBER peut être utilisée conjointement avec différentes valeurs de CONNECTION pour identifier différents éléments de données dans les différentes instances de processus de l'application EDD.

Structure lexicale

COMMAND identifier, [OPERATION, TRANSACTION+, INDEX, BLOCK_B, NUMBER, SLOT, CONNECTION, HEADER, MODULE, RESPONSE_CODES]

Les attributs de COMMAND sont spécifiés au Tableau 43.

Tableau 43 – Attributs de COMMAND

Usage	Attribut	Description
m	OPERATION	Élément lexical (voir 7.7.2.1)
m	TRANSACTION	Élément lexical (voir 7.7.2.2)
c	INDEX	Élément lexical (voir 7.7.2.3)
s	BLOCK_B	Élément lexical (voir 7.7.2.4)
s	NUMBER	Élément lexical (voir 7.7.2.5)
s	SLOT	Élément lexical (voir 7.7.2.6)
o	CONNECTION	Élément lexical (voir 7.7.2.7)
o	HEADER	Élément lexical (voir 7.7.2.8)
o	MODULE	Élément lexical (voir 7.7.2.9)
o	RESPONSE_CODES	Élément lexical (voir 7.30.5)

NOTE 1 Les codes de réponse apparaissant dans une transaction s'appliquent uniquement à cette transaction.

NOTE 2 Les codes de réponse apparaissant à l'extérieur d'une transaction s'appliquent à toutes les transactions.

NOTE 3 Si le même code de réponse est spécifié à la fois à l'intérieur et à l'extérieur d'une transaction, le code de réponse spécifié dans la transaction est prioritaire, mais uniquement pour cette transaction.

7.7.2 Attributs spécifiques

7.7.2.1 OPERATION

Objet

L'attribut OPERATION spécifie les actions effectuées par l'appareil de terrain lors de la réception d'une demande de service provenant du système de communication.

Structure lexicale

OPERATION [(COMMAND, DATA_EXCHANGE, READ, string, WRITE)<exp>]

Les attributs d'OPERATION sont spécifiés au Tableau 44.

Tableau 44 – Attributs d'OPERATION

Usage	Attribut	Description
s	COMMAND	Spécifie que l'appareil effectue un ensemble prédéfini d'actions (auto-essai, réinitialisation principale, par exemple). Les actions à effectuer sont spécifiques à l'appareil et sont identifiées par l'attribut NUMBER.
s	DATA_EXCHANGE	Spécifie une transaction bidirectionnelle. Les données d'entrée sont spécifiées dans l'attribut REPLY et les données de sortie sont spécifiées dans l'attribut REQUEST de la TRANSACTION.
s	READ	Spécifie une transaction de lecture. L'appareil de terrain renvoie les valeurs en cours des variables spécifiées. VARIABLES doit être défini dans la section REPLY de TRANSACTION.
s	string	Spécifie les services spécifiques au fabricant pour les transactions. Il convient d'expliquer la transaction à exécuter dans une spécification correspondante.
s	WRITE	Spécifie une transaction d'écriture. L'appareil de terrain reçoit les valeurs en cours des variables spécifiées. VARIABLES doit être défini dans la section REQUEST de TRANSACTION.

7.7.2.2 TRANSACTION

7.7.2.2.1 Structure générale

Objet

Les transactions spécifient le champ de données des messages de demande et de réponse d'une commande. Plus d'une transaction peut être définie. Il convient que la syntaxe prenne en charge une identification unique de la TRANSACTION. Chaque TRANSACTION peut comprendre un ensemble de codes de réponse qui s'applique seulement à cette transaction particulière.

EXEMPLE Les commandes de bloc, telles que les commandes 4 et 5 de HART³ Universal Command version 4 ou supérieure, sont des exemples de commandes de transaction multiples.

Structure lexicale

TRANSACTION [REPLY, REQUEST, integer, RESPONSE_CODES]

Les attributs de TRANSACTION sont spécifiés au Tableau 45.

³ Voir les quatre caractères PACKED_ASCII

Tableau 45 – Attributs de TRANSACTION

Usage	Attribut	Description
m	REPLY	Élément lexical (voir 7.7.2.2.2.1)
m	REQUEST	Élément lexical (voir 7.7.2.2.2.2)
o	integer	Spécifie le nombre de TRANSACTIONS (si plusieurs objets TRANSACTION sont utilisés)
o	RESPONSE_CODES	Élément lexical (voir 7.24)

7.7.2.2.2 Attributs spécifiques**7.7.2.2.2.1 REPLY****Objet**

L'attribut REPLY contient une liste d'éléments de données (constantes ou références à des variables) qui sont reçus depuis l'appareil. L'ordre de la liste doit être maintenu dans le champ de données correspondant dans le PDU communiqué.

Lorsqu'une VARIABLE ne commence et ne termine pas sur des limites d'octet, il convient qu'un masque soit utilisé pour spécifier comment la variable est encapsulée dans le champ de données.

Chaque bit dans le masque peut avoir une signification sémantique. De plus, si le bit le moins significatif est défini dans un masque d'élément, la donnée suivante (le cas échéant) est contenue dans l'octet(s) suivant(s) du PDU. Si le bit le moins significatif est nul (non défini), l'élément de données suivant est contenu dans le(s) même(s) octet(s) de la PDU.

L'attribut item-mask

NOTE 1 Même si le bit le moins significatif n'est pas utilisé pour des données réelles, il convient de le définir dans un masque d'élément (associé à un élément de données "fictif") si des éléments de données suivent.

NOTE 2 Des exemples sont présentés à l'Annexe C.

Structure lexicale

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

Les attributs de REPLY sont spécifiés au Tableau 46.

Tableau 46 – Attributs de REPLY et REQUEST

Usage	Attribut	Description
m	reference	Référence à une instance de VARIABLE
o	item-mask	Spécifie un motif de bits pour extraire la VARIABLE du champ de données Le masque d'élément doit être utilisé uniquement pour les types de données INTEGER, UNSIGNED INTEGER, ENUMERATED et BIT_ENUMERATED. Si le bit le moins significatif est défini dans item-mask, l'élément suivant de la liste REQUEST ou REPLY doit être utilisé La longueur d'item-mask doit être de 1 octet à 4 octets et doit correspondre à la longueur de l'élément de données dans la liste REQUEST ou REPLY
o	INFO	Spécifie que la VARIABLE n'est pas réellement stockée dans l'appareil. Elle contient des informations supplémentaires pour assurer un traitement correct
o	INDEX	Spécifie que la VARIABLE est utilisée dans REQUEST ou REPLY en tant qu'index dans un REFERENCE_ARRAY

NOTE 3 Une variable peut être qualifiée à la fois par INDEX et INFO, auquel cas elle est appelée variable d'index local. Les commandes avec des indices locaux se comportent exactement de la même façon que les commandes avec des variables qualifiées par INDEX, sauf que les variables d'index ne sont pas stockées dans l'appareil.

NOTE 4 INFO peut être utilisé pour envoyer ou lire une VARIABLE avec des unités différentes de celles utilisées dans l'appareil.

7.7.2.2.2 REQUEST

Objet

L'attribut REQUEST contient une liste d'éléments de données (constantes ou références à des variables) qui sont envoyés à l'appareil. L'ordre de la liste doit être maintenu dans le champ de données correspondant dans le PDU communiqué.

Si un élément de données ne commence et ne termine pas sur des limites d'octet du PDU, un masque d'élément doit être utilisé pour spécifier comment l'élément de données est encapsulé dans le champ de données. Le masque d'élément est un nombre entier d'octets qui doit être logiquement soumis à une opération ET avec le PDU communiqué pour extraire l'élément de données souhaité.

Si aucun masque de donnée n'est spécifié, un masque implicite est utilisé avec tous les bits définis pour chaque octet, correspondant à la longueur de donnée.

Chaque bit dans le masque peut avoir une signification sémantique. De plus, si le bit le moins significatif est défini dans un masque d'élément, la donnée suivante (le cas échéant) est contenue dans l'octet(s) suivant(s) du PDU. Si le bit le moins significatif est nul (non défini), l'élément de données suivant est contenu dans le(s) même(s) octet(s) de la PDU.

NOTE 1 Même si le bit le moins significatif n'est pas utilisé pour des données réelles, il convient de le définir dans un masque d'élément (associé à un élément de données "fictif") si des éléments de données suivent.

NOTE 2 Des exemples sont présentés à l'Annexe C.

Structure lexicale

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

Les attributs de REQUEST sont spécifiés au Tableau 46.

7.7.2.3 INDEX

Objet

L'attribut INDEX doit être présent si l'attribut SLOT (tranche) ou BLOCK_B (bloc B) est utilisé. L'INDEX spécifie l'adresse d'élément de données dans le SLOT ou le BLOCK_B.

Structure lexicale

INDEX (integer, reference, expression)<exp>

Les attributs d'INDEX sont spécifiés au Tableau 47.

Tableau 47 – Attribut d'INDEX

Usage	Attribut	Description
s	integer	Numéro de l'index
s	reference	Référence à une instance de VARIABLE contenant l'index
s	expression	Expression qui doit être évaluée à un entier non négatif compris dans l'intervalle d'index de l'attribut INDEX (pour la description des expressions, voir 7.34.7).

7.7.2.4 BLOCK_B

Objet

L'attribut BLOCK_B fournit une référence à un bloc individuel dans l'appareil.

Structure lexicale

BLOCK_B (reference) <exp>

L'attribut de BLOCK_B est spécifié au Tableau 48.

Tableau 48 – Attribut de BLOCK_B

Usage	Attribut	Description
m	reference	Référence à une instance de BLOCK_B

7.7.2.5 NUMBER

Objet

L'attribut NUMBER (numéro) est utilisé pour énumérer les objets COMMAND (commande).

Structure lexicale

NUMBER (integer, reference, expression) <exp>

Les attributs de NUMBER sont spécifiés au Tableau 49.

Tableau 49 – Attribut de NUMBER

Usage	Attribut	Description
s	integer	Numéro de COMMAND
s	reference	Référence à une instance de VARIABLE contenant le numéro de COMMAND.
s	expression	Expression qui doit être évaluée à un entier non négatif qui est dans l'intervalle d'index de l'attribut NUMBER (pour la description des expressions, voir 7.34.7).

7.7.2.6 SLOT

Objet

Les données d'un dispositif peuvent être allouées à des plages logiques. Dans une plage, chaque élément de données est adressé à l'aide d'un index. L'attribut SLOT spécifie le numéro de la plage.

Structure lexicale

SLOT (integer, reference, expression) <exp>

Les attributs de SLOT sont spécifiés au Tableau 50.

Tableau 50 – Attribut de SLOT

Usage	Attribut	Description
s	integer	Numéro de la plage
s	reference	Référence à une instance de VARIABLE contenant le numéro de plage
s	expression	Expression qui doit être évaluée à un entier non négatif compris dans l'intervalle d'index de l'attribut SLOT (pour la description des expressions, voir 7.34.7).

7.7.2.7 CONNECTION

Objet

L'attribut CONNECTION (connexion) contient une référence à la construction de base CONNECTION qui contient le numéro d'une instance du processus d'application de l'EDD.

Structure lexicale

CONNECTION (reference) <exp>

L'attribut de CONNECTION est spécifié au Tableau 51.

Tableau 51 – Attribut de CONNECTION

Usage	Attribut	Description
m	reference	Référence à une instance de CONNECTION spécifiant l'application EDD.

7.7.2.8 HEADER

Objet

L'attribut HEADER (en-tête) est utilisé pour des protocoles de communication spécifiques.

NOTE Il peut être utilisé pour adresser des données dans un appareil ou pour une opération de réinitialisation.

Structure lexicale

HEADER (string) <exp>

L'attribut de HEADER est spécifié au Tableau 52.

Tableau 52 – Attribut HEADER

Usage	Attribut	Description
m	string	Spécifie une chaîne utilisée pour la communication avec des protocoles de communication spécifiques

7.7.2.9 MODULE

Objet

L'attribut MODULE donne une référence externe. L'utilisation de cet attribut est hors du domaine d'application de la présente norme.

NOTE Pour des raisons de compatibilité, le MODULE est inclus avec des appareils existants utilisant d'autres fichiers de description.

Structure lexicale

MODULE (reference) <exp>

L'attribut de MODULE est spécifié au Tableau 53.

Tableau 53 – Attribut de MODULE

Usage	Attribut	Description
m	reference	Référence à une instance de MODULE dans un fichier de description externe

7.8 CONNECTION

7.8.1 Structure générale

Objet

Un dispositif peut prendre en compte plusieurs applications EDD. Chaque application EDD est représentée sous la forme d'une instance de processus d'application EDD. La construction de base CONNECTION fournit une référence à une instance de processus d'application EDD d'un appareil.

Le nom associé à une construction CONNECTION peut être utilisé dans l'attribut CONNECTION de COMMAND.

Structure lexicale

CONNECTION *identifier*, APPINSTANCE

L'attribut de CONNECTION est spécifié au Tableau 54.

Tableau 54 – Attribut de CONNECTION

Usage	Attribut	Description
m	APPINSTANCE	Élément lexical (voir 7.8.2)

7.8.2 Attribut spécifique – APPINSTANCE

Objet

Plusieurs applications EDD peuvent être définies dans un dispositif. Chaque application EDD représente une instance de processus de l'application EDD. Les instances de processus de l'application EDD permettent différents niveaux d'accès.

Structure lexicale

APPINSTANCE (*integer*, *reference*)<exp>

Les attributs d'APPINSTANCE sont spécifiés au Tableau 55.

Tableau 55 – Attributs d'APPINSTANCE

Usage	Attribut	Description
s	integer	Numéro de l'instance de processus d'application EDD
s	reference	Référence à une instance de VARIABLE contenant le nombre d'instances de processus d'application EDD

7.9 DOMAIN

7.9.1 Structure générale

Objet

Un DOMAIN (domaine) représente un espace mémoire dans un dispositif. Il peut contenir des programmes ou des données. L'identifiant spécifie le DOMAIN correspondant de l'appareil. Les données, programmes (qui sont alloués dans le DOMAIN), taille, accès, etc., sont spécifiés à l'extérieur de l'EDD.

NOTE 1 DOMAIN peut être utilisé pour transférer de grandes quantités de données vers et depuis un appareil.

NOTE 2 Afin de pouvoir transmettre ou télécharger DOMAIN, il est nécessaire que l'appareil prenne en charge les services spécifiques au DOMAIN.

Structure lexicale

DOMAIN identifier [HANDLING, RESPONSE_CODE]

Les attributs de DOMAIN sont spécifiés au Tableau 56.

Tableau 56 – Attributs de DOMAIN

Usage	Attribut	Description
o	HANDLING	Élément lexical (voir 7.9.2.1)
o	RESPONSE_CODES	Élément lexical (voir 7.30.5)

7.9.2 Attributs spécifiques

7.9.2.1 HANDLING

Objet

L'attribut HANDLING spécifie les opérations qui peuvent être effectuées sur le DOMAIN. Par défaut, un DOMAIN sans attribut HANDLING peut être lu et écrit.

Structure lexicale

HANDLING (READ, READ_WRITE, WRITE) <exp>

Les attributs de HANDLING sont spécifiés au Tableau 57.

Tableau 57 – Attribut de HANDLING

Usage	Attribut	Description
s	READ	Indique que le contenu de la mémoire de DOMAIN peut uniquement être lu depuis l'appareil
s	READ_WRITE	Indique que le contenu de la mémoire de DOMAIN peut uniquement être lu depuis et écrit dans l'appareil
s	WRITE	Indique que le contenu de la mémoire de DOMAIN peut uniquement être écrit dans l'appareil

7.9.2.2 Généralités

DOMAIN ne comporte qu'un seul attribut spécifique.

7.10 EDIT_DISPLAY

7.10.1 Structure générale

Objet

Un élément EDIT_DISPLAY (édition-affichage) définit comment les données sont gérées pour l'affichage et l'édition. Il est utilisé pour grouper des éléments conjointement pour édition.

NOTE Cette construction est incluse pour la rétrocompatibilité. Pour de futures implémentations, la construction de base plus générale de MENU est recommandée.

Structure lexicale

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

Les attributs d'EDIT_DISPLAY sont spécifiés au Tableau 58.

Tableau 58 – Attributs d'EDIT_DISPLAY

Usage	Attribut	Description
M	EDIT_ITEMS	Élément lexical (voir 7.10.2.1)
M	LABEL	Élément lexical (voir 7.30.3)
O	DISPLAY_ITEMS	Élément lexical (voir 7.10.2.2)
O	POST_EDIT_ACTIONS	Élément lexical (voir 7.10.2.3)
O	PRE_EDIT_ACTIONS	Élément lexical (voir 7.10.2.4)

7.10.2 Attributs spécifiques

7.10.2.1 EDIT_ITEMS

Objet

EDIT_ITEMS définit l'ensemble des éléments de données qui doivent être présentés à l'utilisateur, et peuvent être édités par l'utilisateur: VARIABLES, relations WRITE_AS_ONE, paramètres de BLOCK_A, et éléments de paramètres de BLOCK_A (c'est-à-dire, les champs ou éléments de PARAMETERS de BLOCK_A de type RECORD (enregistrement) ou VALUE_ARRAY (matrice de valeurs)).

NOTE 1 Les détails spécifiques pour la présentation et l'édition d'un élément de données (valeur par défaut, facteur d'échelle, intervalle, par exemple) sont spécifiés dans les définitions de chaque élément individuel référencé.

NOTE 2 Si une relation WRITE_AS_ONE apparaît en tant qu'élément EDIT_ITEM, il convient que l'utilisateur examine toutes les variables de cette relation. L'utilisateur peut ne pas modifier toutes les valeurs mais il convient au moins qu'il valide que les valeurs actuelles sont acceptables.

Structure lexicale

EDIT_ITEMS [(reference) <exp>] +

L'attribut d'EDIT_ITEMS est spécifié au Tableau 59.

Tableau 59 – Attribut d'EDIT_ITEMS

Usage	Attribut	Description
m	reference	Référence à une instance de VARIABLE, WRITE_AS_ONE, BLOCK_A PARAMETERS ou à un élément de BLOCK_A PARAMETERS. Les références autorisées sont spécifiques au profil.

7.10.2.2 DISPLAY_ITEMS

Objet

DISPLAY_ITEMS définit l'ensemble des éléments de données qui doivent être présentés à l'utilisateur pour information, c'est-à-dire que ceux-ci ne doivent pas être édités: VARIABLES, BLOCK_A PARAMETERS, et éléments de BLOCK_A PARAMETERS (c'est-à-dire, respectivement des champs ou des éléments de BLOCK_A PARAMETERS de type RECORD ou VALUE_ARRAY).

NOTE Les détails spécifiques pour la présentation d'un élément de données (valeur par défaut, facteur d'échelle, intervalle, par exemple) sont spécifiés dans les définitions de chaque élément individuel référencé.

Structure lexicale

DISPLAY_ITEMS [(reference) <exp>] +

L'attribut de DISPLAY_ITEMS est spécifié au Tableau 60.

Tableau 60 – Attributs de DISPLAY_ITEM

Usage	Attribut	Description
m	reference	Référence à une instance de VARIABLE, WRITE_AS_ONE, BLOCK_A PARAMETERS ou à un élément de BLOCK_A PARAMETERS. Les références autorisées sont spécifiques au profil.

7.10.2.3 POST_EDIT_ACTIONS

Objet

L'attribut POST_EDIT_ACTIONS (actions post-édition) spécifie les METHOD qui doivent être exécutées lorsque l'utilisateur a fini le traitement de l'objet EDIT_DISPLAY (édition/affichage). Si l'EDIT_DISPLAY est abandonné, les POST_EDIT_ACTIONS ne sont pas exécutées. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une méthode se termine anormalement, les METHOD suivantes ne sont pas exécutées.

Les POST_EDIT_ACTIONS éventuelles d'une entité référencée dans les EDIT_ITEMS (éléments pour édition) doivent être exécutées lorsqu'une nouvelle valeur a été entrée. Les POST_EDIT_ACTIONS d'EDIT_DISPLAY sont exécutées à l'issue de toutes ces POST_EDIT_ACTIONS individuelles uniquement.

Structure lexicale

POST_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de POST_EDIT_ACTIONS sont spécifiés au Tableau 61.

7.10.2.4 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS (actions pré-édition) spécifie les METHOD qui doivent être exécutées immédiatement lorsque l'EDIT_DISPLAY est activé. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une méthode se termine anormalement, les METHOD suivantes ne sont pas exécutées et l'activation d'EDIT_DISPLAY est annulée.

Les PRE_EDIT_ACTIONS d'EDIT_DISPLAY doivent être exécutées avant toute PRE_EDIT_ACTIONS définie des entités référencées.

Structure lexicale

PRE_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de PRE_EDIT_ACTIONS sont spécifiés au Tableau 61.

Tableau 61 – Attribut de POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS

Usage	Attribut	Description
m	reference	Référence à une instance de METHOD
o	DEFINITION	Élément lexical (voir 7.30.1)

7.11 FILE

7.11.1 Structure générale

Objet

FILE (fichier) décrit une zone de stockage persistant qui est gérée par l'application EDD. Chaque dispositif physique a sa propre copie de la zone de stockage, c'est-à-dire un élément

FILE associé à une instance d'un dispositif et non au type d'un dispositif. L'attribut MEMBERS définit la structure du magasin.

Structure lexicale

FILE *identifiant* [MEMBERS, HELP, LABEL]

Les attributs de FILE sont spécifiés au Tableau 62.

Tableau 62 – Attributs de FILE

Usage	Attribut	Description
m	MEMBERS	Élément lexical (voir 7.30.4)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)

7.11.2 Généralités

FILE ne comporte qu'un seul attribut spécifique.

7.12 GRAPH

7.12.1 Structure générale

Objet

GRAPH (graphique) décrit un graphique utilisé pour afficher des données provenant d'un dispositif de l'application EDD. Un GRAPH est utilisé pour afficher un ensemble de données qui est stocké dans un dispositif, par opposition à un CHART (diagramme) qui est utilisé pour afficher des valeurs de données collectées en continu depuis un dispositif.

Structure lexicale

GRAPH *identifiant* [MEMBERS, HEIGHT, WIDTH, CYCLE_TIME, HELP, LABEL, X_AXIS]

Les attributs de GRAPH sont spécifiés au Tableau 63.

Tableau 63 – Attributs de GRAPH

Usage	Attribut	Description
m	MEMBERS	Élément lexical (voir 7.30.4)
o	HEIGHT	Élément lexical (voir 7.12.2.1)
o	WIDTH	Élément lexical (voir 7.12.2.2)
o	CYCLE_TIME	Élément lexical (voir 7.12.2.3)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)
o	X_AXIS	Élément lexical (voir 7.12.2.4)

7.12.2 Attributs spécifiques

7.12.2.1 HEIGHT

Objet

L'attribut HEIGHT spécifie la largeur de la taille verticale qu'il convient que GRAPH ait lors de son affichage. Par défaut, la taille d'un GRAPH sans attribut HEIGHT est MEDIUM.

Structure lexicale

HEIGHT (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

Les attributs de HEIGHT sont spécifiés au Tableau 64.

Tableau 64 – Attribut de HEIGHT/WIDTHs

Usage	Attribut	Description
s	XX_SMALL	Spécifie qu'il convient que GRAPH occupe une partie extrêmement petite de la fenêtre dans laquelle il est affiché
s	X_SMALL	Spécifie qu'il convient que GRAPH occupe une très petite partie de la fenêtre dans laquelle il est affiché
s	SMALL	Spécifie qu'il convient que GRAPH occupe une petite partie de la fenêtre dans laquelle il est affiché
s	MEDIUM	Spécifie qu'il convient que GRAPH occupe une partie moyenne de la fenêtre dans laquelle il est affiché
s	LARGE	Spécifie qu'il convient que GRAPH occupe une grande partie de la fenêtre dans laquelle il est affiché
s	X_LARGE	Spécifie qu'il convient que GRAPH occupe une très grande partie de la fenêtre dans laquelle il est affiché
s	XX_LARGE	Spécifie qu'il convient que GRAPH occupe une partie extrêmement grande de la fenêtre dans laquelle il est affiché

7.12.2.2 WIDTH

Objet

L'attribut WIDTH spécifie la largeur de la taille horizontale qu'il convient que GRAPH ait lors de son affichage. Par défaut, la taille d'un GRAPH sans attribut WIDTH est MEDIUM.

Structure lexicale

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

Les attributs de WIDTH sont spécifiés au Tableau 64.

7.12.2.3 CYCLE_TIME

Objet

L'attribut CYCLE_TIME (durée de cycle) spécifie le temps, en ms, durant lequel l'application EDD doit attendre entre des lectures successives des valeurs sources de l'appareil. Par défaut, un GRAPH n'a pas de durée de cycle.

Structure lexicale

CYCLE_TIME (integer) <exp>

L'attribut de CYCLE_TIME est spécifié au Tableau 65.

Tableau 65 – Attribut de CYCLE_TIME

Usage	Attribut	Description
m	integer	Temps, en ms, entre des lectures successives des valeurs sources de l'appareil.

7.12.2.4 X_AXIS

Objet

L'attribut X_AXIS spécifie le mode d'affichage de l'axe X. Chaque WAVEFORM (courbe) sur un GRAPH partage un X_AXIS commun mais chaque WAVEFORM sur un GRAPH a son propre Y_AXIS (axe Y). Par défaut, l'affichage de l'axe X d'un GRAPH sans attribut X_AXIS est dérivé des WAVEFORM sur le GRAPH.

Structure lexicale

X_AXIS (reference) <exp>

L'attribut de X_AXIS est spécifié au Tableau 66.

Tableau 66 – Attribut de X_AXIS

Usage	Attribut	Description
m	reference	Référence à une instance d'AXIS

7.13 GRID

7.13.1 Structure générale

Objet

GRID (grille) décrit une matrice de données provenant d'un dispositif qui sont destinées à être affichées par l'application EDD. Une GRID est utilisée pour afficher des vecteurs de données avec l'en-tête ou la description des données dans ce vecteur. Les vecteurs sont affichés horizontalement (lignes) ou verticalement (colonnes) comme spécifié par l'attribut ORIENTATION.

Structure lexicale

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

Les attributs de GRID sont spécifiés au Tableau 67.

Tableau 67 – Attributs de GRID

Usage	Attribut	Description
m	VECTORS	Élément lexical (voir 7.13.2.1)
o	HANDLING	Élément lexical (voir 7.13.2.2)
o	HEIGHT	Élément lexical (voir 7.13.2.3)
o	WIDTH	Élément lexical (voir 7.13.2.4)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)
o	ORIENTATION	Élément lexical (voir 7.13.2.5)
o	VALIDITY	Élément lexical (voir 7.13.2.6)

7.13.2 Attributs spécifiques

7.13.2.1 VECTORS

Objet

L'attribut VECTORS (vecteurs) spécifie le contenu de l'élément GRID. Chaque vecteur est affiché dans l'ordre. Pour une GRID verticale, les vecteurs sont affichés de gauche à droite.

Pour une GRID horizontale, les vecteurs sont affichés de haut en bas. Chaque vecteur est constitué d'une description qui constitue une étiquette pour le vecteur suivi par les données à afficher.

Structure lexicale

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

Les attributs de VECTORS sont spécifiés au Tableau 68.

Tableau 68 – Attributs de VECTORS

Usage	Attribut	Description
m	description	Chaîne contenant une brève description des données contenues dans le vecteur
o	reference	Référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY, un élément d'une instance de REFERENCE_ARRAY, une instance de VALUE_ARRAY ou une instance de REFERENCE_ARRAY
	string	Chaîne constante ou référence représentée par une chaîne
o	double	Constante à virgule flottante à double précision
o	float	Constante à virgule flottante à simple précision
o	integer	Constante entière
o	unsigned_integer	Constante entière non signée

7.13.2.2 HANDLING

Objet

L'attribut HANDLING spécifie les opérations qui peuvent être effectuées sur les données du GRID. Par défaut, les données d'un GRID sans attribut HANDLING peuvent être lues et écrites.

Structure lexicale

HANDLING (READ, READ_WRITE, WRITE) <exp>

Les attributs de HANDLING sont spécifiés au Tableau 69.

Tableau 69 – Attribut de HANDLING

Usage	Attribut	Description
s	READ	Spécifie que les données du GRID peuvent être lues
s	READ_WRITE	Spécifie que les données du GRID peuvent être lues et écrites
s	WRITE	Spécifie que les données du GRID peuvent être écrites

7.13.2.3 HEIGHT

Objet

L'attribut HEIGHT spécifie la largeur de la taille verticale qu'il convient que GRID ait lors de son affichage. Par défaut, la taille d'un GRID sans attribut HEIGHT est MEDIUM.

Structure lexicale

HEIGHT (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

Les attributs de HEIGHT sont spécifiés au Tableau 70.

Tableau 70 – Attributs de HEIGHT/WIDTH

Usage	Attribut	Description
s	XX_SMALL	Spécifie qu'il convient que GRID occupe une partie extrêmement petite de la fenêtre dans laquelle il est affiché
s	X_SMALL	Spécifie qu'il convient que GRID occupe une très petite partie de la fenêtre dans laquelle il est affiché
s	SMALL	Spécifie qu'il convient que GRID occupe une petite partie de la fenêtre dans laquelle il est affiché.
s	MEDIUM	Spécifie qu'il convient que GRID occupe une partie moyenne de la fenêtre dans laquelle il est affiché.
s	LARGE	Spécifie qu'il convient que GRID occupe une grande partie de la fenêtre dans laquelle il est affiché.
s	X_LARGE	Spécifie qu'il convient que GRID occupe une très grande partie de la fenêtre dans laquelle il est affiché
s	XX_LARGE	Spécifie qu'il convient que GRID occupe une partie extrêmement grande de la fenêtre dans laquelle il est affiché

7.13.2.4 WIDTH**Objet**

L'attribut WIDTH spécifie la largeur de la taille horizontale qu'il convient que GRID ait lors de son affichage. Par défaut, la taille d'un GRID sans attribut WIDTH est MEDIUM.

Structure lexicale

WIDTH (XX_SMALL, X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE, XX_LARGE) <exp>

Les attributs de WIDTH sont spécifiés au Tableau 70.

7.13.2.5 ORIENTATION**Objet**

L'attribut ORIENTATION spécifie la direction selon laquelle les vecteurs sont affichés. Par défaut, une GRID sans attribut ORIENTATION est affichée verticalement.

Structure lexicale

ORIENTATION (HORIZONTAL, VERTICAL) <exp>

Les attributs d'ORIENTATION sont spécifiés au Tableau 71.

Tableau 71 – Attribut d'ORIENTATION

Usage	Attribut	Description
s	HORIZONTAL	Spécifie que les vecteurs doivent être affichés dans des lignes
s	VERTICAL	Spécifie que les vecteurs doivent être affichés dans des colonnes

7.13.2.6 VALIDITY**Objet**

L'attribut VALIDITY spécifie si un GRID est ou n'est pas valide. Si un GRID n'est pas valide, il ne s'affiche pas. Par défaut, un GRID sans attribut VALIDITY est toujours valide.

NOTE Il convient d'exprimer VALIDITY à l'aide d'une instruction conditionnelle (IF, IF-ELSE, SELECT).

Structure lexicale

VALIDITY [(FALSE, TRUE)<exp>]

Les attributs de VALIDITY sont spécifiés au Tableau 72.

Tableau 72 – Attributs de VALIDITY

Usage	Attribut	Description
s	FALSE	Spécifie que l'instance de GRID n'est pas valide
s	TRUE	Spécifie que l'instance de GRID est valide

7.14 IMAGE

7.14.1 Structure générale

Objet

IMAGE décrit une image visuelle.

Structure lexicale

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

Les attributs d'IMAGE sont spécifiés au Tableau 73.

Tableau 73 – Attributs d'IMAGE

Usage	Attribut	Description
m	PATH	Élément lexical (voir 7.14.2.1)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)
o	LINK	Élément lexical (voir 7.14.2.2)
o	VALIDITY	Élément lexical (voir 7.14.2.3)

7.14.2 Attributs spécifiques

7.14.2.1 PATH

Objet

L'attribut PATH spécifie le nom du fichier de l'image. Des images destinées à plusieurs environnements locaux peuvent être spécifiées grâce à la même technique de localisation que celle utilisée avec les chaînes littérales.

EXEMPLE “[en|english.jpeg|de|german.jpeg]”

Structure lexicale

PATH (string)<exp>

L'attribut de PATH est spécifié au Tableau 74.

Tableau 74 – Attribut de PATH

Usage	Attribut	Description
m	string	Spécifie le nom de fichier de l'image.

7.14.2.2 LINK

Objet

L'attribut LINK spécifie un MENU ou une METHOD qui peut être accessible depuis l'IMAGE. Cela permet d'obtenir des informations additionnelles concernant l'IMAGE.

Structure lexicale

LINK (reference) <exp>

L'attribut de LINK est spécifié au Tableau 75.

Tableau 75 – Attribut de LINK

Usage	Attribut	Description
m	reference	Référence à une instance de MENU ou une instance de METHOD

7.14.2.3 VALIDITY

Objet

L'attribut VALIDITY spécifie si une IMAGE est ou n'est pas valide. Si une IMAGE n'est pas valide, il convient de ne pas l'afficher. Une IMAGE sans attribut VALIDITY est toujours valide.

NOTE Il convient d'exprimer VALIDITY à l'aide d'une instruction conditionnelle (IF, IF-ELSE, SELECT).

Structure lexicale

VALIDITY [(FALSE, TRUE) <exp>]

Les attributs de VALIDITY sont spécifiés au Tableau 76.

Tableau 76 – Attributs de VALIDITY

Usage	Attribut	Description
s	FALSE	Spécifie que l'instance d'IMAGE n'est pas valide
s	TRUE	Spécifie que l'instance d'IMAGE est valide

7.15 IMPORT

7.15.1 Structure générale

Objet

Les constructions de langage EDDL précédemment spécifiées dans cet article sont suffisantes pour décrire un dispositif individuel quelconque. Cependant, des mécanismes additionnels sont nécessaires pour décrire des révisions multiples d'un dispositif individuel ou décrire des parties d'un dispositif générique. Pour produire ces mécanismes, une EDD peut importer une autre EDD. L'EDD importée peut elle-même importer également d'autres EDD.

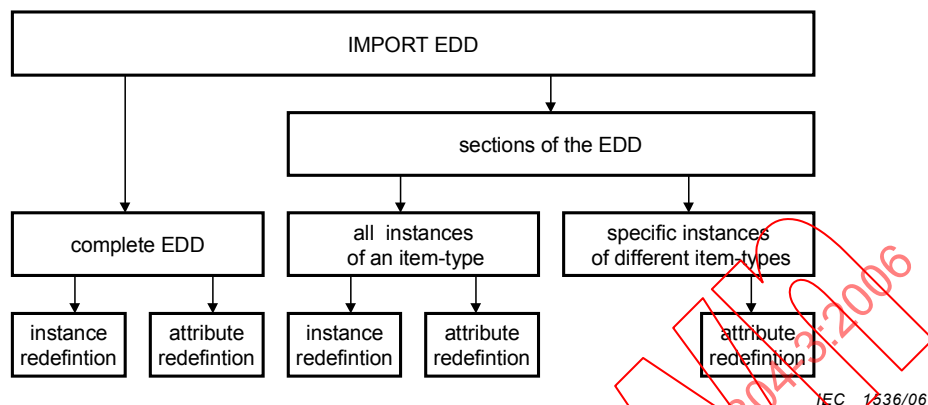
NOTE Avec ces mécanismes, l'EDD d'une nouvelle version de l'appareil peut être spécifiée en important simplement l'EDD de l'ancienne version de l'appareil et en spécifiant les modifications d'éléments. Ce type d'EDD est appelé description delta, car l'EDD entière est spécifiée par des modifications d'une EDD existante.

Trois types d'importation doivent être pris en charge:

- importation de tous les éléments, où tous les éléments de l'EDD externe sont importés;
- importation des éléments d'un type spécifié, où uniquement les éléments des types spécifiés sont importés. Si un type spécifié est importé, des importations d'éléments spécifiques du même type ne sont pas autorisées;

- importation d'un élément spécifique.

L'identifiant d'un élément EDD importé ne doit pas être modifié. La Figure 27 illustre les mécanismes d'importation d'EDDL.



Légende

Anglais	Français
sections of the EDD	sections de l'EDD
complete EDD	EDD complet
all instances of an item-type	toutes les instances d'un item-type
specific instances of different item-types	instances particulières de différents item-types
instance redefinition	redéfinition d'instance
attribute redefinition	redéfinition d'attribut

Figure 27 – Mécanismes d'importation EDDL

Une autre méthode d'importation consiste à référencer l'EDD par le nom du fichier.

Structure lexicale

```

IMPORT ((DD_REVISION, DEVICE_REVISION, DEVICE_TYPE, EDD_PROFILE,
EDD_VERSION, MANUFACTURER, MANUFACTURER_EXT), FILE_NAME),
(EVERYTHING,
(AXES, BLOCKS_A, BLOCKS_B, CHART, COLLECTIONS, COMMANDS, CONNECTIONS,
DOMAINS, EDIT_DISPLAYS, FILES, GRAPHS, LISTS, MENUS, METHODS, PROGRAMS,
RECORDS, REFERENCE_ARRAYS, REFRESHES, RELATIONS, RESPONSE_CODES, SOURCES,
UNITS, VALUE_ARRAYS, VARIABLES, VARIABLE_LISTS, WAVEFORMS, WRITE_AS_ONES,
reference*,
[DELETE identifier]*,
[REDEFINE identifier]*,
attribute-redefinition*))
  
```

Les attributs sont spécifiés au Tableau 77.

Tableau 77 – Description d'importation d'un appareil

Usage	Attribut	Description
o	attribute-redefinition	(voir 7.15.2)
o	AXES	Spécifie que toutes les instances d'AXIS d'une EDD externe sont importées
o	BLOCKS_A	Spécifie que toutes les instances de BLOCK_A d'une EDD externe sont importées
o	BLOCKS_B	Spécifie que toutes les instances de BLOCK_B d'une EDD externe sont importées
o	CHARTS	Spécifie que toutes les instances de CHART d'une EDD externe sont importées
o	COLLECTIONS	Spécifie que toutes les instances de COLLECTION d'une EDD externe sont importées
o	COMMANDS	Spécifie que toutes les instances de COMMAND d'une EDD externe sont importées
o	CONNECTIONS	Spécifie que toutes les instances de CONNECTION d'une EDD externe sont importées
m	DD_REVISION	(voir 7.2.2.1)
o	DELETE	Spécifie qu'une instance importée doit être supprimée
m	DEVICE_REVISION	(voir 7.2.2.2)
m	DEVICE_TYPE	(voir 7.2.2.3)
o	DOMAINS	Spécifie que toutes les instances de DOMAIN d'une EDD externe sont importées
o	EDD_PROFILE	(voir 7.2.2.4)
o	EDD_VERSION	(voir 7.2.2.5)
o	EDIT_DISPLAYS	Spécifie que toutes les instances d'EDIT_DISPLAY d'une EDD externe sont importées
o	EVERYTHING	Spécifie que tous les éléments d'une EDD externe sont importés. Si cette forme est utilisée, aucun type d'élément et aucune instance individuelle ne peuvent être sélectionnés. Les attributs DELETE, REDEFINE et attribute-redefinition peuvent être utilisés
o	FILE_NAME	Spécifie une chaîne contenant un nom de fichier et son chemin d'accès dans une structure de répertoire. Le fichier contient l'EDD qui doit être importée. En utilisant ce type de référencement, la spécification des informations d'identification d'EDD n'est pas nécessaire
o	FILES	Spécifie que toutes les instances de FILE d'une EDD externe sont importées
o	GRAPHS	Spécifie que toutes les instances de GRAPH d'une EDD externe sont importées
o	LISTS	Spécifie que toutes les instances de LIST d'une EDD externe sont importées
m	MANUFACTURER	(voir 7.2.2.6)
c	MANUFACTURER_EXT	(voir 7.2.2.7)
o	MENUS	Spécifie que toutes les instances de MENU d'une EDD externe sont importées
o	METHODS	Spécifie que toutes les instances de METHOD d'une EDD externe sont importées
o	PROGRAMS	Spécifie que toutes les instances de PROGRAM d'une EDD externe sont importées
o	RECORDS	Spécifie que toutes les instances de RECORD d'une EDD externe sont importées
o	REDEFINE	Spécifie que tous les attributs d'une instance importée doivent être redéfinis
o	reference	Spécifie qu'une seule instance est importée. Une seule instance ne doit pas être importée si elle est déjà incluse dans une importation de type d'élément
o	REFERENCE_ARRAYS	Spécifie que toutes les instances de REFERENCE_ARRAY d'une EDD externe sont importées
o	REFRESHES	Spécifie que toutes les instances de REFRESH d'une EDD externe sont importées

Usage	Attribut	Description
o	RELATIONS	Spécifie que toutes les instances de REFRESH, UNIT et WRITE_AS_ONE d'une EDD externe sont importées
o	RESPONSE_CODES	Spécifie que toutes les instances de RESPONSE_CODE d'une EDD externe sont importées
o	SOURCES	Spécifie que toutes les instances de SOURCE d'une EDD externe sont importées
o	UNITS	Spécifie que toutes les instances d'UNIT d'une EDD externe sont importées
o	VALUE_ARRAYS	Spécifie que toutes les instances de VALUE_ARRAY d'une EDD externe sont importées
o	VARIABLE_LISTS	Spécifie que toutes les instances de VARIABLE_LIST d'une EDD externe sont importées
o	VARIABLES	Spécifie que toutes les instances de VARIABLE d'une EDD externe sont importées
o	WAVEFORMS	Spécifie que toutes les instances de WAVEFORM d'une EDD externe sont importées
o	WRITE_AS_ONES	Spécifie que toutes les instances de WRITE_AS_ONE d'une EDD externe sont importées

7.15.2 Attributs spécifiques

7.15.2.1 Redéfinition d'attributs

Objet

Des attributs d'une instance importée peuvent être ajoutés, supprimés ou redéfinis. L'ajout, la suppression et la redéfinition n'affectent pas tous les attributs d'une construction de l'EDD de l'élément de base.

Les attributs de redéfinition sont spécifiés au Tableau 78.

Tableau 78 – Attributs de redéfinition

Usage	Attribut	Description
s	ADD	Ajoute l'attribut spécifié suivant. "A" indique que ADD peut être appliqué à l'attribut dans les tables des règles de redéfinition (voir Tableau 79 à Tableau 103)
s	DELETE	Supprime l'attribut spécifié suivant. "D" indique que DELETE peut être appliqué à l'attribut dans les tables des règles de redéfinition (voir Tableau 79 à Tableau 103)
s	REDEFINE	Redéfinit l'attribut spécifié suivant. "R" indique que REDEFINE peut être appliqué à l'attribut dans les tables des règles de redéfinition (voir Tableau 79 à Tableau 103)

Structure lexicale — AXIS

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

Les règles de redéfinition d'AXIS sont spécifiées au Tableau 78 et au Tableau 79.

Tableau 79 – Règles de redéfinition pour les attributs d'AXIS

A	D	R	Attribut	Description
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	MAX_VALUE	Élément lexical (voir 7.3.2.1)
	•	•	MIN_VALUE	Élément lexical (voir 7.3.2.2)
	•	•	SCALING	Élément lexical (voir 7.3.2.3)

Structure lexicale — BLOCK_A

BLOCK identifier, [[ADD, DELETE, REDEFINE], [CHARACTERISTICS, LABEL, PARAMETERS, parameters-list-element, AXIS_ITEMS, CHART_ITEMS, COLLECTION_ITEMS, EDIT_DISP_ITEMS, FILE_ITEMS, GRAPH_ITEMS, LIST_ITEMS, LOCAL_PARAMETERS, local-parameters-list-element, HELP, MENU_ITEMS, METHOD_ITEMS, PARAMETER_LISTS, parameters-lists-list-element, REFERENCE_ARRAY_ITEMS, REFRESH_ITEMS, SOURCE_ITEMS, UNIT_ITEMS, WAVEFORM_ITEMS, WRITE_AS_ONE_ITEMS]]+

Les règles de redéfinition de BLOCK_A sont spécifiées au Tableau 78 et au Tableau 80.

Tableau 80 – Règles de redéfinition pour les attributs de BLOCK_A

A	D	R	Attribut	Description
		•	CHARACTERISTICS	Élément lexical (voir 7.4.1.2.1)
		•	LABEL	Élément lexical (voir 7.30.3)
		•	PARAMETERS	Élément lexical (voir 7.4.1.2.2)
•	•	•	parameters-list-element	La liste d'énumération des éléments individuels de PARAMETERS peut être modifiée
	•	•	AXIS_ITEMS	Élément lexical (voir 7.4.1.2.3)
	•	•	CHART_ITEMS	Élément lexical (voir 7.4.1.2.4)
	•	•	COLLECTION_ITEMS	Élément lexical (voir 7.4.1.2.5)
	•	•	EDIT_DISP_ITEMS	Élément lexical (voir 7.4.1.2.6)
	•	•	FILE_ITEMS	Élément lexical (voir 7.4.1.2.7)
	•	•	GRAPH_ITEMS	Élément lexical (voir 7.4.1.2.8)
	•	•	GRID_ITEMS	Éléments lexicaux (voir 7.4.1.2.9)
	•	•	IMAGE_ITEMS	Éléments lexicaux (voir 7.4.1.2.10)
	•	•	LIST_ITEMS	Élément lexical (voir 7.4.1.2.11)
		•	LOCAL_PARAMETERS	Élément lexical (voir 7.4.1.2.12)
•	•	•	local-parameters-list-element	La liste d'énumération des éléments individuels de LOCAL_PARAMETERS peut être modifiée
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	MENU_ITEMS	Élément lexical (voir 7.4.1.2.13)
	•	•	METHOD_ITEMS	Élément lexical (voir 7.4.1.2.14)
		•	PARAMETER_LISTS	Élément lexical (voir 7.4.1.2.15)
•	•	•	parameters-lists-list-element	La liste d'énumération des éléments individuels de PARAMETERS_LISTS peut être modifiée
	•	•	REFERENCE_ARRAY_ITEMS	Élément lexical (voir 7.4.1.2.16)
	•	•	REFRESH_ITEMS	Élément lexical (voir 7.4.1.2.17)
	•	•	SOURCE_ITEMS	Élément lexical (voir 7.4.1.2.18)
	•	•	UNIT_ITEMS	Élément lexical (voir 7.4.1.2.19)

A	D	R	Attribut	Description
	•	•	WAVEFORM_ITEMS	Élément lexical (voir 7.4.1.2.20)
	•	•	WRITE_AS_ONE_ITEMS	Élément lexical (voir 7.4.1.2.21)

Structure lexicale – BLOCK_B

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

Les règles de redéfinition de BLOCK_B sont spécifiées au Tableau 78 et au Tableau 81.

Tableau 81 – Règles de redéfinition pour les attributs de BLOCK_B

A	D	R	Attribut	Description
		•	NUMBER	Élément lexical (voir 7.4.2.2.1)
		•	TYPE	Élément lexical (voir 7.4.2.2.2)

Structure lexicale — CHART

CHART identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, CYCLE_TIME, DISPLAY_SIZE, HELP, LABEL, LENGTH, TYPE]]+

Les règles de redéfinition de COLLECTION sont spécifiées au Tableau 78 et au Tableau 82.

Tableau 82 – Règles de redéfinition pour les attributs de CHART

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.30.4)
•	•	•	members-list-element	La liste d'énumération des éléments individuels de MEMBERS peut être modifiée
		•	CYCLE_TIME	Élément lexical (voir 7.5.2.1)
		•	HEIGHT	Élément lexical (voir 7.5.2.2)
		•	WIDTH	Élément lexical (voir 7.5.2.3)
		•	HELP	Élément lexical (voir 7.30.2)
		•	LABEL	Élément lexical (voir 7.30.3)
		•	LENGTH	Élément lexical (voir 7.5.2.4)
		•	TYPE	Élément lexical (voir 7.5.2.5)

Structure lexicale — COLLECTION

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

Les règles de redéfinition de COLLECTION sont spécifiées au Tableau 78 et au Tableau 83.

Tableau 83 – Règles de redéfinition pour les attributs de COLLECTION

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.30.4)
•	•	•	members-list-element	La liste d'énumération des éléments individuels de MEMBERS peut être modifiée
		•	HELP	Élément lexical (voir 7.30.2)
		•	LABEL	Élément lexical (voir 7.30.3)

Structure lexicale — COMMAND

COMMAND *identifieur*, [[ADD, DELETE, REDEFINE], [OPERATION, TRANSACTION, INDEX, BLOCK_B, NUMBER, SLOT, CONNECTION, HEADER, MODULE, RESPONSE_CODES, response-codes-list-element]]+

Les règles de redéfinition de COMMAND sont spécifiées au Tableau 78 et au Tableau 84.

Tableau 84 – Règles de redéfinition pour les attributs de COMMAND

A	D	R	Attribut	Description
		•	OPERATION	Élément lexical (voir 7.7.2.1)
•	•	•	TRANSACTION	Élément lexical (voir 7.7.2.2)
		•	INDEX	Élément lexical (voir 7.7.2.3)
		•	BLOCK_B	Élément lexical (voir 7.7.2.4)
		•	NUMBER	Élément lexical (voir 7.7.2.5)
		•	SLOT	Élément lexical (voir 7.7.2.6)
	•	•	CONNECTION	Élément lexical (voir 7.7.2.7)
	•	•	HEADER	Élément lexical (voir 7.7.2.8)
	•	•	MODULE	Élément lexical (voir 7.7.2.9)
	•	•	RESPONSE_CODES	Élément lexical (voir 7.30.5)
•	•	•	réponse-codes-list-element	La liste d'énumération des éléments individuels RESPONSE_CODE peut être modifiée

Structure lexicale — CONNECTION

CONNECTION *identifieur*, [REDEFINE, [APPINSTANCE]]+

Les règles de redéfinition de CONNECTION sont spécifiées au Tableau 78 et au Tableau 85.

Tableau 85 – Règles de redéfinition pour les attributs de CONNECTION

A	D	R	Attribut	Description
		•	APPINSTANCE	Élément lexical (voir 7.8.2)

Structure lexicale — DOMAIN

DOMAIN *identifieur*, [DELETE, REDEFINE], [HANDLING, RESPONSE_CODES]+

Les règles de redéfinition de DOMAIN sont spécifiées au Tableau 78 et au Tableau 86.

Tableau 86 – Règles de redéfinition pour les attributs de DOMAIN

A	D	R	Attribut	Description
	•	•	HANDLING	Élément lexical (voir 7.9.2)
	•	•	RESPONSE_CODES	Élément lexical (voir 7.30.5)

Structure lexicale — EDIT_DISPLAY

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

Les règles de redéfinition d'EDIT_DISPLAY sont spécifiées au Tableau 78 et au Tableau 87.

Tableau 87 – Règles de redéfinition pour les attributs d'EDIT_DISPLAY

A	D	R	Attribut	Description
		•	EDIT_ITEMS	Élément lexical (voir 7.10.2.1)
		•	LABEL	Élément lexical (voir 7.30.3)
	•	•	DISPLAY_ITEMS	Élément lexical (voir 7.10.2.2)
	•	•	POST_EDIT_ACTIONS	Élément lexical (voir 7.10.2.3)
	•	•	PRE_EDIT_ACTIONS	Élément lexical (voir 7.10.2.4)

Structure lexicale — FILE

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

Les règles de redéfinition de FILE sont spécifiées au Tableau 78 et au Tableau 88.

Tableau 88 – Règles de redéfinition pour les attributs de FILE

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.30.4)
•	•	•	members-list-element	La liste d'énumération des éléments individuels de MEMBERS peut être modifiée
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)

Structure lexicale — GRAPH

GRAPH identifier, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, DISPLAY_SIZE, HELP, LABEL, Y_AXIS]]+

Les règles de redéfinition de GRAPH sont spécifiées au Tableau 78 et au Tableau 89.

Tableau 89 – Règles de redéfinition pour les attributs de GRAPH

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.30.4)
•	•	•	members-list-element	La liste d'énumération des éléments individuels de MEMBERS peut être modifiée
	•	•	HEIGHT	Élément lexical (voir 7.12.2.1)
	•	•	WIDTH	Élément lexical (voir 7.12.2.2)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	CYCLE_TIME	Élément lexical (voir 7.12.2.3)
	•	•	X_AXIS	Élément lexical (voir 7.12.2.4)

Structure lexicale — GRID

GRID identifier, [[ADD, DELETE, REDEFINE], PATH, HELP, LABEL, LINK, VALIDITY]]+

Les règles de redéfinition de GRAPH sont spécifiées au Tableau 78 et au Tableau 90.

Tableau 90 – Règles de redéfinition pour les attributs de GRID

A	D	R	Attribut	Description
		•	VECTORS	Élément lexical (voir 7.13.2.1)
	•	•	HANDLING	Élément lexical (voir 7.13.2.2)
	•	•	HEIGHT	Élément lexical (voir 7.13.2.3)
	•	•	WIDTH	Élément lexical (voir 7.13.2.4)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	ORIENTATION	Élément lexical (voir 7.13.2.5)
	•	•	VALIDITY	Élément lexical (voir 7.13.2.6)

Structure lexicale — IMAGE

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

Les règles de redéfinition de GRAPH sont spécifiées au Tableau 78 et au Tableau 91.

Tableau 91 – Règles de redéfinition pour les attributs d'IMAGE

A	D	R	Attribut	Description
		•	PATH	Élément lexical (voir 7.14.2.1)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	LINK	Élément lexical (voir 7.14.2.2)
	•	•	VALIDITY	Élément lexical (voir 7.14.2.3)

Structure lexicale — LIST

LIST identifieur, [[DELETE, REDEFINE], [TYPE, CAPACITY, COUNT, HELP, LABEL]]+

Les règles de redéfinition de LIST sont spécifiées Tableau 78 et au Tableau 92.

Tableau 92 – Règles de redéfinition pour les attributs de LIST

A	D	R	Attribut	Description
		•	TYPE	Élément lexical (voir 7.17.2.1)
	•	•	CAPACITY	Élément lexical (voir 7.17.2.2)
	•	•	COUNT	Élément lexical (voir 7.17.2.3)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)

Structure lexicale — MENU

MENU identifieur, [[ADD, DELETE, REDEFINE], [ITEMS, items-list-element, LABEL, ACCESS, ENTRY, HELP, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTIONS, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, PURPOSE, ROLE, STYLE, VALIDITY]]+

Les règles de redéfinition de MENU sont spécifiées au Tableau 78 et au Tableau 93.

Tableau 93 – Règles de redéfinition pour les attributs de MENU

A	D	R	Attribut	Description
		•	ITEMS	Élément lexical (voir 7.18.2.1)
•	•	•	items-list-element	La liste d'énumération des éléments individuels d'ITEMS peut être modifiée
		•	LABEL	Élément lexical (voir 7.30.3)
	•	•	ACCESS	Élément lexical (voir 7.18.2.2)
	•	•	ENTRY	Élément lexical (voir 7.18.2.3)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	POST_EDIT_ACTIONS	Élément lexical (voir 7.18.2.4)
	•	•	POST_READ_ACTIONS	Élément lexical (voir 7.18.2.5)
	•	•	POST_WRITE_ACTIONS	Élément lexical (voir 7.18.2.6)
	•	•	PRE_EDIT_ACTIONS	Élément lexical (voir 7.18.2.7)
	•	•	PRE_READ_ACTIONS	Élément lexical (voir 7.18.2.8)
	•	•	PRE_WRITE_ACTIONS	Élément lexical (voir 7.18.2.9)
	•	•	PURPOSE	Élément lexical (voir 7.18.2.10)
	•	•	ROLE	Élément lexical (voir 7.18.2.11)
	•	•	STYLE	Élément lexical (voir 7.18.2.12)
	•	•	VALIDITY	Élément lexical (voir 7.18.2.13)

Structure lexicale — METHOD

METHOD *identif*ier, [[DELETE, REDEFINE], [ACCESS, CLASS, DEFINITION, LABEL, HELP, VALIDITY]]+

Les règles de redéfinition de METHOD sont spécifiées au Tableau 78 et au Tableau 94.

Tableau 94 – Règles de redéfinition pour les attributs de METHOD

A	D	R	Attribut	Description
	•	•	ACCESS	Élément lexical (voir 7.19.2.1)
		•	CLASS	Élément lexical (voir 7.19.2.2)
		•	DEFINITION	Élément lexical (voir 7.30.1)
		•	LABEL	Élément lexical (voir 7.30.3)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	VALIDITY	Élément lexical (voir 7.19.2.3)

Structure lexicale — PROGRAM

PROGRAM *identif*ier, [[DELETE, REDEFINE], [ARGUMENT, argument-list-element, RESPONSE_CODES]]+

Les règles de redéfinition de PROGRAM sont spécifiées au Tableau 78 et au Tableau 95.

Tableau 95 – Règles de redéfinition pour les attributs de PROGRAM

A	D	R	Attribut	Description
		•	ARGUMENT	Élément lexical (voir 7.20.2)
	•	•	argument-list-element	La liste d'énumération des éléments individuels d'ARGUMENTS peut être modifiée
	•	•	RESPONSE_CODES	Élément lexical (voir 7.30.5)

Structure lexicale — RECORD

RECORD *identifieur*, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, LABEL, HELP, RESPONSE_CODES]]+

Les règles de redéfinition de RECORD sont spécifiées au Tableau 78 et au Tableau 96.

Tableau 96 – Règles de redéfinition pour les attributs de RECORD

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.30.4)
•	•	•	members-list-element	La liste d'énumération des éléments individuels de MEMBERS peut être modifiée
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	RESPONSE_CODES	Élément lexical (voir 7.30.5)

Structure lexicale — REFERENCE_ARRAY

REFERENCE_ARRAY *identifieur*, [[ADD, DELETE, REDEFINE], [ELEMENTS, elements-list-element, HELP, LABEL]]+

Les règles de redéfinition de REFERENCE_ARRAY sont spécifiées au Tableau 78 et au Tableau 97.

Tableau 97 – Règles de redéfinition pour les attributs de REFERENCE_ARRAY

A	D	R	Attribut	Description
		•	ELEMENTS	Élément lexical (voir 7.22.2)
•	•	•	elements-list-element	La liste d'énumération des éléments individuels d'ELEMENTS peut être modifiée
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	HELP	Élément lexical (voir 7.30.2)

Structure lexicale — RESPONSE_CODES

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

Les règles de redéfinition de RESPONSE_CODES sont spécifiées au Tableau 78 et au Tableau 98.

Tableau 98 – Règles de redéfinition pour les attributs de RESPONSE_CODES

A	D	R	Attribut	Description
•	•	•	response-code-list-element	La liste d'énumération des éléments individuels RESPONSE_CODE peut être modifiée

Structure lexicale — SOURCE

SOURCE *identifieur*, [[ADD, DELETE, REDEFINE], [MEMBERS, members-list-element, EMPHASIS, HELP, LABEL, LINE_TYPE, Y_AXIS]]+

Les règles de redéfinition de SOURCE sont spécifiées au Tableau 78 et au Tableau 99.

Tableau 99 – Règles de redéfinition pour les attributs de SOURCE

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.30.4)
•	•	•	members-list-element	La liste d'énumération des éléments individuels de MEMBERS peut être modifiée
	•	•	EMPHASIS	Élément lexical (voir 7.25.2.1)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	LINE_COLOR	Élément lexical (voir 7.25.2.2)
	•	•	LINE_TYPE	Élément lexical (voir 7.25.2.3)
	•	•	Y_AXIS	Élément lexical (voir 7.25.2.4)
	•	•	EXIT ACTIONS	Élément lexical (voir 7.25.2.5)
	•	•	INIT_ACTIONS	Élément lexical (voir 7.25.2.6)
	•	•	REFRESH_ACTIONS	Élément lexical (voir 7.25.2.7)

Structure lexicale — VALUE_ARRAY

VALUE_ARRAY *identifieur*, [[DELETE, REDEFINE], [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, RESPONSE_CODES]]+

Les règles de redéfinition de VALUE_ARRAY sont spécifiées au Tableau 78 et au Tableau 100.

Tableau 100 – Règles de redéfinition pour les attributs de VALUE_ARRAY

A	D	R	Attribut	Description
		•	LABEL	Élément lexical (voir 7.30.3)
		•	NUMBER_OF_ELEMENTS	Élément lexical (voir 7.26.2.1)
		•	TYPE	Élément lexical (voir 7.26.2.2)
•	•	•	HELP	Élément lexical (voir 7.30.2)
•	•	•	RESPONSE_CODES	Élément lexical (voir 7.30.5)

Structure lexicale — VARIABLE

VARIABLE *identifieur*, [[ADD, DELETE, REDEFINE], [CLASS, LABEL, TYPE, DEFAULT_VALUE, DISPLAY_FORMAT, EDIT_FORMAT, INITIAL_VALUE, MAX_VALUE, MIN_VALUE, SCALING_FACTOR, type-list-element, CONSTANT_UNIT, HANDLING, HELP, POST_EDIT_ACTIONS, POST_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_EDIT_ACTION, PRE_READ_ACTIONS, PRE_WRITE_ACTIONS, READ_TIMEOUT, RESPONSE_CODES, STYLE, VALIDITY, WRITE_TIMEOUT]]+

Les règles de redéfinition sont spécifiées au Tableau 78 et au Tableau 101.

Tableau 101 – Règles de redéfinition pour les attributs de VARIABLE

A	D	R	Attribut	Description
		•	CLASS	Élément lexical (voir 7.27.2.1)
		•	LABEL	Élément lexical (voir 7.30.3)
			TYPE	Élément lexical (voir 7.27.2.2)
•	•	•	DEFAULT_VALUE	Élément lexical (voir 7.27.2.2)
•	•	•	DISPLAY_FORMAT	Élément lexical (voir 7.27.2.2)
•	•	•	EDIT_FORMAT	Élément lexical (voir 7.27.2.2)

A	D	R	Attribut	Description
	•	•	INITIAL_VALUE	Élément lexical (voir 7.27.2.2)
	•	•	MAX_VALUE	Élément lexical (voir 7.27.2.2)
	•	•	MIN_VALUE	Élément lexical (voir 7.27.2.2)
	•	•	SCALING_FACTOR	Élément lexical (voir 7.27.2.2)
•	•	•	type-list-element	La liste d'énumération des éléments individuels de chaque TYPE peut être modifiée
	•	•	CONSTANT_UNIT	Élément lexical (voir 7.27.2.3)
	•	•	HANDLING	Élément lexical (voir 7.27.2.4)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	POST_EDIT_ACTIONS	Élément lexical (voir 7.27.2.6)
	•	•	POST_READ_ACTIONS	Élément lexical (voir 7.27.2.8)
	•	•	POST_WRITE_ACTIONS	Élément lexical (voir 7.27.2.9)
	•	•	PRE_EDIT_ACTION	Élément lexical (voir 7.27.2.10)
	•	•	PRE_READ_ACTIONS	Élément lexical (voir 7.27.2.11)
	•	•	PRE_WRITE_ACTIONS	Élément lexical (voir 7.27.2.12)
	•	•	REFRESH_ACTIONS	Élément lexical (voir 7.27.2.13)
	•	•	READ_TIMEOUT	Élément lexical (voir 7.27.2.13)
	•	•	RESPONSE_CODES	Élément lexical (voir 7.30.5)
	•	•	STYLE	Élément lexical (voir 7.27.2.15)
	•	•	VALIDITY	Élément lexical (voir 7.27.2.16)
	•	•	WRITE_TIMEOUT	Élément lexical (voir 7.27.2.13)

Structure lexicale — VARIABLE_LIST

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

Les règles de redéfinition de VARIABLE_LIST sont spécifiées au Tableau 78 et au Tableau 102.

Tableau 102 – Règles de redéfinition pour les attributs de VARIABLE_LIST

A	D	R	Attribut	Description
		•	MEMBERS	Élément lexical (voir 7.30.4)
•	•	•	members-list-element	La liste d'énumération des éléments individuels de MEMBERS peut être modifiée.
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	RESPONSE_CODES	Élément lexical (voir 7.30.5)

Structure lexicale — WAVEFORM

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_TYPE, REFRESH_ACTIONS, Y_AXIS]]+

Les règles de redéfinition de WAVEFORM sont spécifiées au Tableau 78 et au Tableau 103.

Tableau 103 – Règles de redéfinition pour les attributs de WAVEFORM

A	D	R	Attribut	Description
		•	TYPE	Élément lexical (voir 7.29.2.1)
•		•	X_VALUES	Élément lexical (voir 7.29.2.1)
•		•	Y_VALUES	Élément lexical (voir 7.29.2.1)
	•	•	NUMBER_OF_POINTS	Élément lexical (voir 7.29.2.1)
		•	X_INITIAL	Élément lexical (voir 7.29.2.1)
		•	X_INCREMENT	Élément lexical (voir 7.29.2.1)
	•	•	EMPHASIS	Élément lexical (voir 7.29.2.2)
	•	•	EXIT_ACTIONS	Élément lexical (voir 7.29.2.3)
	•	•	HANDLING	Élément lexical (voir 7.29.2.3)
	•	•	HELP	Élément lexical (voir 7.30.2)
	•	•	INIT_ACTIONS	Élément lexical (voir 7.29.2.5)
•	•	•	KEY_POINTS	Élément lexical (voir 7.29.2.6)
	•	•	LABEL	Élément lexical (voir 7.30.3)
	•	•	LINE_COLOR	Élément lexical (voir 7.29.2.7)
	•	•	LINE_TYPE	Élément lexical (voir 7.29.2.8)
	•	•	REFRESH_ACTIONS	Élément lexical (voir 7.29.2.9)
	•	•	Y_AXIS	Élément lexical (voir 7.29.2.10)

7.16 LIKE

Objet

LIKE (similaire) est utilisé pour créer une nouvelle instance d'EDD sur la base d'un élément existant. LIKE fait une copie du type d'élément existant avec tous les attributs.

Structure lexicale

LIKE *identifieur*, *reference*, *attribute-redefinition*+, *item-type*

Les attributs sont spécifiés au Tableau 104.

Tableau 104 – Attributs de LIKE

Usage	Attribut	Description
m	identifieur	Nom de la nouvelle instance d'EDD
m	reference	Référence à une instance d'EDD qui doit être copiée
o	attribute-redefinition	Spécifie si un attribut est ajouté, supprimé ou redéfini (voir 7.15.2). Seuls les attributs existants peuvent être supprimés ou redéfinis
o	item-type	Peut être un des éléments énumérés au Tableau 42. Même si item-type est présent, il n'est pas interprété

7.17 LIST

7.17.1 Structure générale

Objet

LIST (liste) décrit une séquence d'éléments dans laquelle chaque élément de la séquence est défini par une instance de construction EDDL.

Structure lexicale

LIST identifier [TYPE, CAPACITY, COUNT, HELP, LABEL]

Les attributs de LIST sont spécifiés au Tableau 105.

Tableau 105 – Attributs de LIST

Usage	Attribut	Description
m	TYPE	Élément lexical (voir 7.17.2.1)
o	CAPACITY	Élément lexical (voir 7.17.2.2)
o	COUNT	Élément lexical (voir 7.17.2.3)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)

7.17.2 Attributs spécifiques

7.17.2.1 TYPE

Objet

L'attribut TYPE est une référence à une instance d'une 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 de TYPE est spécifié au Tableau 106.

Tableau 106 – Attribut de TYPE

Usage	Attribut	Description
m	reference	Référence à une instance de VARIABLE, une instance de RECORD, une instance de VALUE_ARRAY, une instance de COLLECTION ou une instance de REFERENCE_ARRAY

7.17.2.2 CAPACITY

Objet

L'attribut CAPACITY (capacité) spécifie le nombre maximal d'éléments qui peuvent être stockés dans la liste. Lorsqu'une liste fait partie d'un FILE (fichier), la CAPACITY ne doit pas être spécifiée étant donné que la capacité est limitée uniquement par les ressources de l'application EDD.

Structure lexicale

CAPACITY (unsigned_integer) <exp>

L'attribut de CAPACITY est spécifié au Tableau 107.

Tableau 107 – Attribut de CAPACITY, COUNT

Usage	Attribut	Description
m	unsigned_integer	Constante entière non signée

7.17.2.3 COUNT

Objet

L'attribut COUNT (compteur) spécifie les nombres d'éléments actuellement stockés dans la liste. Lorsqu'une liste fait partie d'un FILE (fichier), le COUNT ne doit pas être spécifié étant donné que seule l'application EDD connaît le nombre d'éléments actuellement présents dans la liste.

Si COUNT est spécifié, il définit la taille de l'instance de LIST lors de sa création. S'il n'est pas défini, une instance de LIST avec zéro élément est créée.

Structure lexicale

COUNT (unsigned_integer) <exp>

L'attribut de COUNT est spécifié au Tableau 107.

7.18 MENU

7.18.1 Structure générale

Objet

MENU organise dans une structure hiérarchique des variables, des méthodes et d'autres éléments spécifiés dans le langage EDDL. Une application utilisateur peut utiliser les ITEMS (options) de MENU pour afficher et entrer des informations d'une façon organisée et cohérente. Les derniers éléments d'une structure de MENU hiérarchique doivent être des méthodes, des variables ou des constructions de base référant leurs groupes (COLLECTION OF VARIABLES, par exemple).

Si un sous-menu s'affiche, le menu dans lequel il est référencé doit être valide.

Structure lexicale

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

Les attributs de MENU sont spécifiés au Tableau 108.

Tableau 108 – Attributs de MENU

Usage	Attribut	Description
m	ITEMS	Élément lexical (voir 7.18.2.1)
m	LABEL	Élément lexical (voir 7.30.3)
o	ACCESS	Élément lexical (voir 7.18.2.2)
o	ENTRY	Élément lexical (voir 7.18.2.3)
o	HELP	Élément lexical (voir 7.30.2)
o	POST_EDIT_ACTIONS	Élément lexical (voir 7.18.2.4)
o	POST_READ_ACTIONS	Élément lexical (voir 7.18.2.5)
o	POST_WRITE_ACTIONS	Élément lexical (voir 7.18.2.6)
o	PRE_EDIT_ACTIONS	Élément lexical (voir 7.18.2.7)
o	PRE_READ_ACTIONS	Élément lexical (voir 7.18.2.8)
o	PRE_WRITE_ACTIONS	Élément lexical (voir 7.18.2.9)
c	PURPOSE	Élément lexical (voir 7.18.2.10). PURPOSE doit être spécifié si l'attribut d'ENTRY est utilisé
o	ROLE	Élément lexical (voir 7.18.2.11)
o	STYLE	Élément lexical (voir 7.18.2.12)
o	VALIDITY	Élément lexical (voir 7.18.2.13)

7.18.2 Attributs spécifiques

7.18.2.1 ITEMS

Objet

L'attribut ITEMS spécifie les éléments sélectionnés (VARIABLE, BLOCK_A PARAMETERS, les éléments de BLOCK_A PARAMETERS, COLLECTION, EDIT_DISPLAY, METHOD, RECORDS, REFERENCE_ARRAY, VALUE_ARRAY, VARIABLE_LIST, GRAPH, CHART et autres MENU) de l'EDD qui doivent s'afficher, ainsi que les identifiants facultatifs de chaque élément. Ces qualificatifs (DISPLAY_VALUE, HIDDEN, READ_ONLY, NO_LABEL, NO_UNIT, REVIEW) sont utilisés pour contrôler le traitement des éléments.

NOTE 1 Les éléments de MENU sont présentés à l'utilisateur dans l'ordre dans lequel ils apparaissent.

NOTE 2 Pour des menus verticaux, le premier élément s'affiche en haut et le dernier élément en bas. Pour des menus horizontaux, le premier élément est affiché sur la gauche et le dernier élément sur la droite.

Structure lexicale

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

Les attributs d'ITEMS sont spécifiés au Tableau 109.

Tableau 109 – Attributs d'ITEMS

Usage	Attribut	Description
m	reference	Référence à une VARIABLE, un BLOCK_A PARAMETERS, un élément de BLOCK_A PARAMETERS, un CHART, une COLLECTION, un EDIT_DISPLAY, un GRAPH, un GRID, une instance d'IMAGE, une instance de MENU, une METHOD, un RECORDS, un REFERENCE_ARRAY, un VALUE_ARRAY ou un VARIABLE_LIST
c/o	DISPLAY_VALUE	"c" est uniquement utilisé avec VARIABLE; "o" est utilisé dans tous les autres cas. Spécifie que la valeur de la VARIABLE correspondante est affichée en plus de son LABEL
o	HIDDEN	Spécifie que l'élément n'est pas affiché
c/o	READ_ONLY	"c" est uniquement utilisé avec VARIABLE; "o" est utilisé dans tous les autres cas. Spécifie que la valeur doit être affichée mais qu'elle ne peut pas être modifiée (quel que soit son attribut HANDLING)
c/o	NO_LABEL	"c" est uniquement utilisé avec VARIABLE; "o" est utilisé dans tous les autres cas. Spécifie que l'étiquette de la VARIABLE n'est pas affichée; DISPLAY_VALUE est implicite
c/o	NO_UNIT	"c" est uniquement utilisé avec VARIABLE; "o" est utilisé dans tous les autres cas. Spécifie que l'unité technique de la VARIABLE n'est pas affichée; DISPLAY_VALUE est implicite
c/o	REVIEW	"c" est uniquement utilisé avec MENU; "o" est utilisé dans tous les autres cas. REVIEW est utilisé pour présenter une grande quantité de données (pour les passer en revue, par exemple)
c/o	INLINE	"c" est uniquement utilisé avec IMAGE; "o" est utilisé dans tous les autres cas. Si l'IMAGE est dans un MENU avec un attribut STYLE égal à WINDOW, DIALOG, PAGE ou GROUP et le qualificatif INLINE n'est pas spécifié, il convient que l'IMAGE couvre l'étendue du MENU; sinon, il convient que l'IMAGE soit affichée en ligne avec les autres ITEMS du MENU
o	SEPARATOR	Spécifie qu'il convient de séparer les éléments de menu avant et après le SEPARATOR (par une ligne horizontale, par exemple)
c/o	COLUMNBREAK	"c" est uniquement utilisé dans un MENU dont l'attribut STYLE est WINDOW, DIALOG, PAGE, ou GROUP; spécifie qu'il convient qu'une nouvelle colonne commence
c/o	ROWBREAK	"c" est uniquement utilisé dans un MENU dont l'attribut STYLE est WINDOW, DIALOG, PAGE, ou GROUP; spécifie qu'il convient qu'une nouvelle ligne commence
o	string	Spécifie qu'il convient d'afficher un texte statique dans le menu

7.18.2.2 ACCESS

Objet

L'attribut ACCESS spécifie si les éléments affichés ou utilisés dans le MENU sont lus depuis l'appareil (ONLINE) ou localement (OFFLINE). Si aucun attribut ACCESS n'est défini, ONLINE est la valeur par défaut.

Structure lexicale

ACCESS [(OFFLINE, ONLINE) <exp>]

Les attributs d'ACCESS sont spécifiés au Tableau 110.

Tableau 110 – Attribut d'ACCESS

Usage	Attribut	Description
s	OFFLINE	Spécifie que les valeurs des éléments affichés dans le MENU sont lues localement
s	ONLINE	Spécifie qu'il convient de lire les valeurs des éléments de MENU dans l'appareil et de les réactualiser à un intervalle approprié

NOTE La mise en œuvre permet de décider à quel moment il convient de transférer les valeurs modifiées vers un appareil (en principe après l'exécution d'une entrée unique ou d'un MENU dans le cas d'un attribut ONLINE ACCESS, ou grâce à un menu séparé pour mettre à jour l'appareil).

7.18.2.3 ENTRY

Objet

L'attribut ENTRY indique si un MENU représente un point d'entrée dans une structure hiérarchique. Si un MENU est défini en tant que point d'entrée, il doit également contenir l'attribut PURPOSE.

Structure lexicale

ENTRY (ROOT) <exp>

L'attribut d'ENTRY est spécifié au Tableau 111.

Tableau 111 – Attribut d'ENTRY

Usage	Attribut	Description
m	ROOT	Spécifie que le MENU peut faire office de point d'entrée

7.18.2.4 POST_EDIT_ACTIONS

Objet

L'attribut POST_EDIT_ACTIONS (actions post-édition) spécifie les METHOD qui doivent être exécutées après que l'utilisateur a fini le traitement du MENU. Si le MENU est abandonné, les POST_EDIT_ACTIONS ne sont pas exécutées. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHOD suivantes ne sont pas exécutées.

Les POST_EDIT_ACTIONS éventuellement associées à une entité référencée dans les ITEMS doivent être exécutées lorsqu'une nouvelle valeur est entrée pour l'entité. Les POST_EDIT_ACTIONS du MENU sont exécutées uniquement après que toutes les POST_EDIT_ACTIONS spécifiques à l'entité sont terminées.

Structure lexicale

POST_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de POST_EDIT_ACTIONS sont spécifiés au Tableau 112.

Tableau 112 – Attributs de POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS

Usage	Attribut	Description
o	reference	Référence à une instance de METHOD
o	DEFINITION	Élément lexical (voir 7.30.1)

7.18.2.5 POST_READ_ACTIONS

Objet

L'attribut POST_READ_ACTIONS (actions post-lecture) spécifie les METHOD qui doivent être exécutées après que toutes les entités référencées dans l'attribut ITEMS ont été lues et traitées. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHOD suivantes ne sont pas exécutées et l'activation de MENU est annulée.

Les POST_READ_ACTIONS éventuellement associées à une entité référencée dans les ITEMS doivent être exécutées lorsqu'une nouvelle valeur est lue pour l'entité. Les POST_READ_ACTIONS du MENU doivent être exécutées uniquement après que toutes les POST_READ_ACTIONS spécifiques à l'entité soient terminées.

Structure lexicale

POST_READ_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de POST_READ_ACTIONS sont spécifiés au Tableau 112.

7.18.2.6 POST_WRITE_ACTIONS

Objet

L'attribut POST_WRITE_ACTIONS (actions post-écriture) spécifie les METHOD qui doivent être exécutées après que toutes les entités référencées dans l'attribut ITEMS ont été écrites et traitées. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHOD suivantes ne sont pas exécutées.

Les POST_WRITE_ACTIONS éventuellement associées à une entité référencée dans les ITEMS doivent être exécutées lorsqu'une nouvelle valeur est écrite pour l'entité. Les POST_WRITE_ACTIONS du MENU doivent être exécutées uniquement après que toutes les POST_WRITE_ACTIONS spécifiques à l'entité soient terminées.

Structure lexicale

POST_WRITE_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de POST_WRITE_ACTIONS sont spécifiés au Tableau 112.

7.18.2.7 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS (actions pré-édition) spécifie les METHOD qui doivent être exécutées immédiatement après l'activation du MENU. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHOD suivantes ne sont pas exécutées et l'activation de MENU est annulée.

Les PRE_EDIT_ACTIONS du MENU doivent être exécutées avant l'une quelconque des PRE_EDIT_ACTIONS définies des entités référencées. Une fois que les PRE_EDIT_ACTIONS de MENU sont terminées, les PRE_EDIT_ACTIONS éventuelles pour les entités référencées doivent être exécutées.

Structure lexicale

PRE_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de PRE_EDIT_ACTIONS sont spécifiés au Tableau 112.

7.18.2.8 PRE_READ_ACTIONS

Objet

L'attribut PRE_READ_ACTIONS (actions prélecture) spécifie les METHOD qui doivent être exécutées avant que toutes les entités référencées dans l'attribut ITEMS soient lues ou traitées. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHOD suivantes ne sont pas exécutées et l'activation de MENU est annulée. Une fois que les

PRE_READ_ACTIONS de MENU sont terminées, les PRE_READ_ACTIONS éventuelles pour les entités référencées doivent être exécutées.

Structure lexicale

PRE_READ_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de PRE_READ_ACTIONS sont spécifiés au Tableau 112.

7.18.2.9 PRE_WRITE_ACTIONS

Objet

L'attribut PRE_WRITE_ACTIONS (actions pré-écriture) spécifie les METHOD qui doivent être exécutées avant que toutes les entités référencées dans l'attribut ITEMS soient écrites ou traitées. Les METHOD spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHOD suivantes ne sont pas exécutées. Une fois que les PRE_WRITE_ACTIONS de MENU sont terminées, les PRE_WRITE_ACTIONS pour les entités référencées doivent être exécutées.

Structure lexicale

PRE_WRITE_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de PRE_WRITE_ACTIONS sont spécifiés au Tableau 112.

7.18.2.10 PURPOSE

Objet

L'attribut PURPOSE (objet) définit l'objet d'application du MENU. Celui-ci indique comment le MENU est utilisé, par exemple, pour une présentation tabulaire, des actions de diagnostic et une liste de transfert/téléchargement.

Structure lexicale

PURPOSE [(CATALOG, DIAGNOSE, LOAD_TO_APPLICATION, LOAD_TO_DEVICE, MENU, PROCESS_VALUE, SIMULATION, string, TABLE)<exp>]+

Les attributs de PURPOSE sont spécifiés au Tableau 113.

Tableau 113 – Attribut de PURPOSE

Usage	Attribut	Description
s	CATALOG	Spécifie que les éléments de l'attribut ITEMS d'une instance de MENU sont des références à des informations spécifiques au catalogue (une variable d'EDD contenant le numéro d'ordre, par exemple)
s	DIAGNOSE	Spécifie que le MENU est utilisé pour des actions de diagnostic
s	LOAD_TO_APPLICATION	Spécifie que le MENU est utilisé pour répertorier les VARIABLES dont les valeurs sont envoyées de l'appareil vers une application
s	LOAD_TO_DEVICE	Spécifie que le MENU est utilisé pour répertorier les VARIABLES dont les valeurs sont envoyées à l'appareil
s	MENU	Spécifie un MENU contenant d'autres éléments de MENU ou éléments de contrôle normalement présentés en utilisant des lignes horizontales
s	PROCESS_VALUE	Spécifie que le MENU est utilisé pour afficher des valeurs de processus
s	SIMULATION	Spécifie que le MENU est utilisé pour des actions de simulation
s	string	Spécifie un PURPOSE défini par l'utilisateur pour le MENU
s	TABLE	Spécifie un MENU contenant d'autres éléments de MENU ou des VARIABLES normalement présentés en utilisant des colonnes verticales

NOTE Il convient d'utiliser l'attribut PURPOSE de MENU pour une représentation de données et de fonction horizontale (des touches de fonction, des menus d'application, par exemple). Il convient qu'un élément TABLE soit utilisé pour une représentation des données d'une fonction verticale (par exemple, des tableaux).

7.18.2.11 ROLE

Objet

L'attribut ROLE (rôle) spécifie dans quel contexte de rôle le MENU est utilisé. Si un MENU contient des sous-menus, seuls ces sous-menus qui ont le même ROLE sont affichés. Si un MENU n'a pas d'attribut ROLE, le MENU doit être affiché lorsqu'il est référencé.

Structure lexicale

ROLE [(string)<exp>]+

L'attribut de ROLE est spécifié au Tableau 114.

Tableau 114 – Attribut de ROLE

Usage	Attribut	Description
m	string	Spécifie une chaîne qui indique le ROLE du MENU

NOTE Une EDD peut contenir un ou plusieurs ROLE suivants: MAINTENANCE, SPECIALIST, DIAGNOSE, HANDHELD, SERVICE, MANUFACTURER, OBSERVER.

7.18.2.12 STYLE

Objet

L'attribut STYLE spécifie la manière dont le MENU est affiché.

Si l'attribut de STYLE est DIALOG, WINDOW, PAGE ou GROUP, le qualificatif DISPLAY_VALUE (voir 7.18.2.1) est implicite dans chaque élément de MENU.

Si un ou plusieurs MENU sont actifs et un MENU DIALOG (boîte de dialogue) est activé, le MENU DIALOG est prioritaire et doit être fermé avant que tout autre MENU puisse être utilisé.

Structure lexicale

STYLE [(DIALOG, WINDOW, PAGE, GROUP, MENU, TABLE, string)<exp>]

Les attributs de STYLE sont spécifiés au Tableau 115.

Tableau 115 – Attribut de STYLE

Usage	Attribut	Description
s	DIALOG	Spécifie qu'aucun autre MENU ne peut être activé pendant que ce MENU est actif. Le MENU doit être fermé avant que des interactions avec d'autres MENUS soient possibles. En d'autres termes, spécifie qu'il convient d'afficher les éléments du MENU sous forme de boîte de dialogue modale
s	WINDOW	Spécifie que d'autres MENUS peuvent être activés pendant que ce MENU est actif Les interactions avec tous les MENUS activés ayant des attributs de STYLE WINDOW sont possibles. En d'autres termes, spécifie qu'il convient d'afficher les éléments du MENU sous forme de fenêtre non modale
s	PAGE	Spécifie qu'il convient d'afficher les éléments du MENU sous forme de page de propriétés à onglets
s	GROUP	Spécifie qu'il convient d'afficher les éléments du MENU sous forme de cadre de groupe
s	MENU	Spécifie qu'il convient d'afficher les éléments du MENU sous forme de menu déroulant ou contextuel

Usage	Attribut	Description
s	TABLE	Spécifie qu'il convient d'afficher les éléments du MENU dans des colonnes verticales
s	string	Désigne un style de MENU spécifique à un outil Si un outil ne peut pas interpréter la chaîne, cet attribut STYLE doit être ignoré

7.18.2.13 VALIDITY

Objet

L'attribut VALIDITY spécifie si un MENU est ou n'est pas valide. Si l'attribut VALIDITY n'est pas défini, la valeur par défaut est TRUE.

NOTE VALIDITY est en principe exprimé à l'aide d'une expression conditionnelle (voir 7.32).

Structure lexicale

VALIDITY [(FALSE, TRUE) <exp>]

Les attributs de VALIDITY sont spécifiés au Tableau 116.

Tableau 116 – Attributs de VALIDITY

Usage	Attribut	Description
s	FALSE	Spécifie que le MENU n'est pas affiché
s	TRUE	Spécifie que le MENU est affiché

7.18.3 Schémas de séquence pour les actions

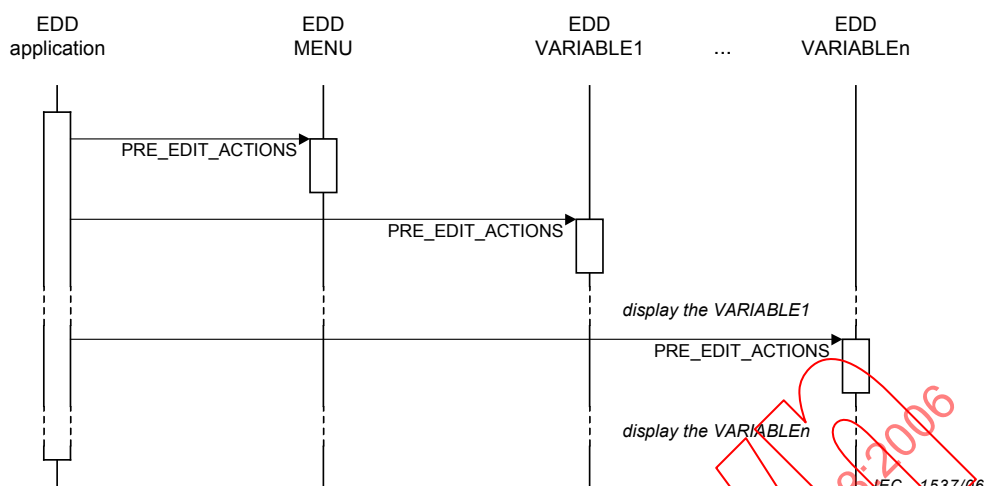
De la Figure 28 à la Figure 34, des schémas de séquence sont présentés pour des actions associées aux MENUS et leurs VARIABLES et METHODS contenues. Les actions sont exécutées et terminées en séquence du haut vers le bas du schéma. Si une action échoue ou est abandonnée pour une raison imprévue, les actions restantes ne sont pas exécutées et le MENU est annulé.

Lorsque le MENU a un ACCESS OFFLINE (hors connexion, données provenant d'une base de données locale), les actions pré/post-lecture/écriture pour la VARIABLE ne sont pas exécutées.

Lorsque le MENU a un ACCESS ONLINE (données lues depuis ou écrites vers l'appareil), les actions pré/post-lecture/écriture pour la VARIABLE doivent être exécutées dans l'ordre indiqué dans la séquence.

Les actions pré/post-édition doivent toujours être exécutées en cas d'ACCESS OFFLINE et ONLINE.

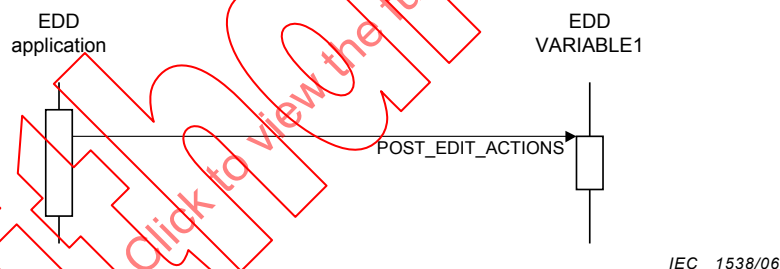
Dans le cas de l'accès en ligne, les valeurs de VARIABLE peuvent être modifiées et immédiatement écrites sauf si elles sont impliquées dans une relation WRITE_AS_ONE. Les entités WRITE_AS_ONE éventuelles sont écrites ensemble. Une fois les VARIABLES WRITE_AS_ONE écrites, les POST_WRITE_ACTIONS sont exécutées.



Légende

Anglais	Français
EDD application	Application EDD
display the VARIABLE1	afficher la VARIABLE1
display the VARIABLEn	afficher la VARIABLEn

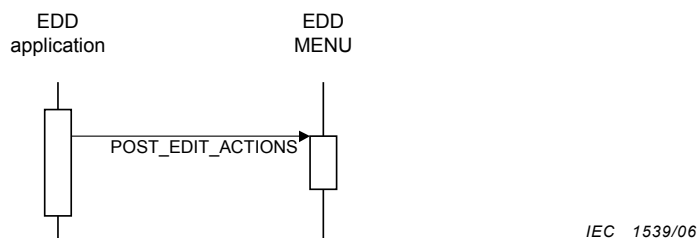
Figure 28 – Activation de MENU (ACCESS OFFLINE)



Légende

Anglais	Français
EDD application	Application EDD

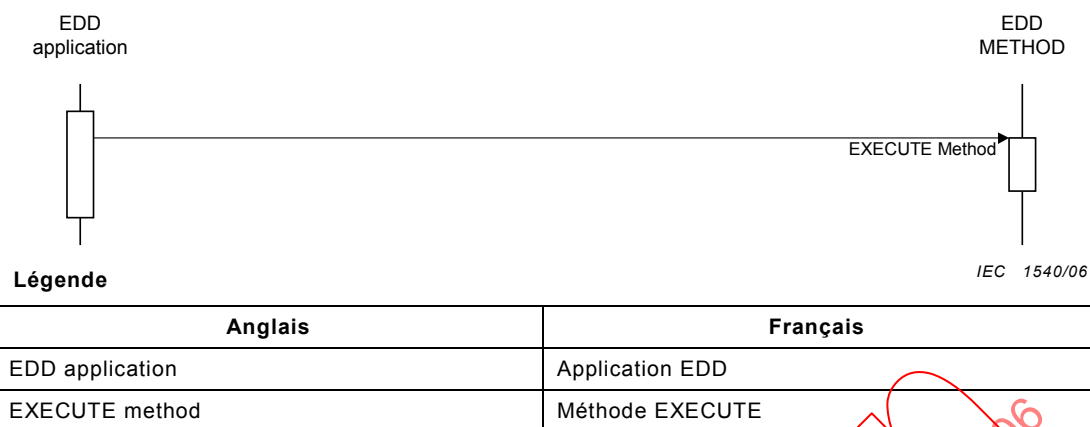
Figure 29 – Action réalisée après l'entrée d'une nouvelle valeur



Légende

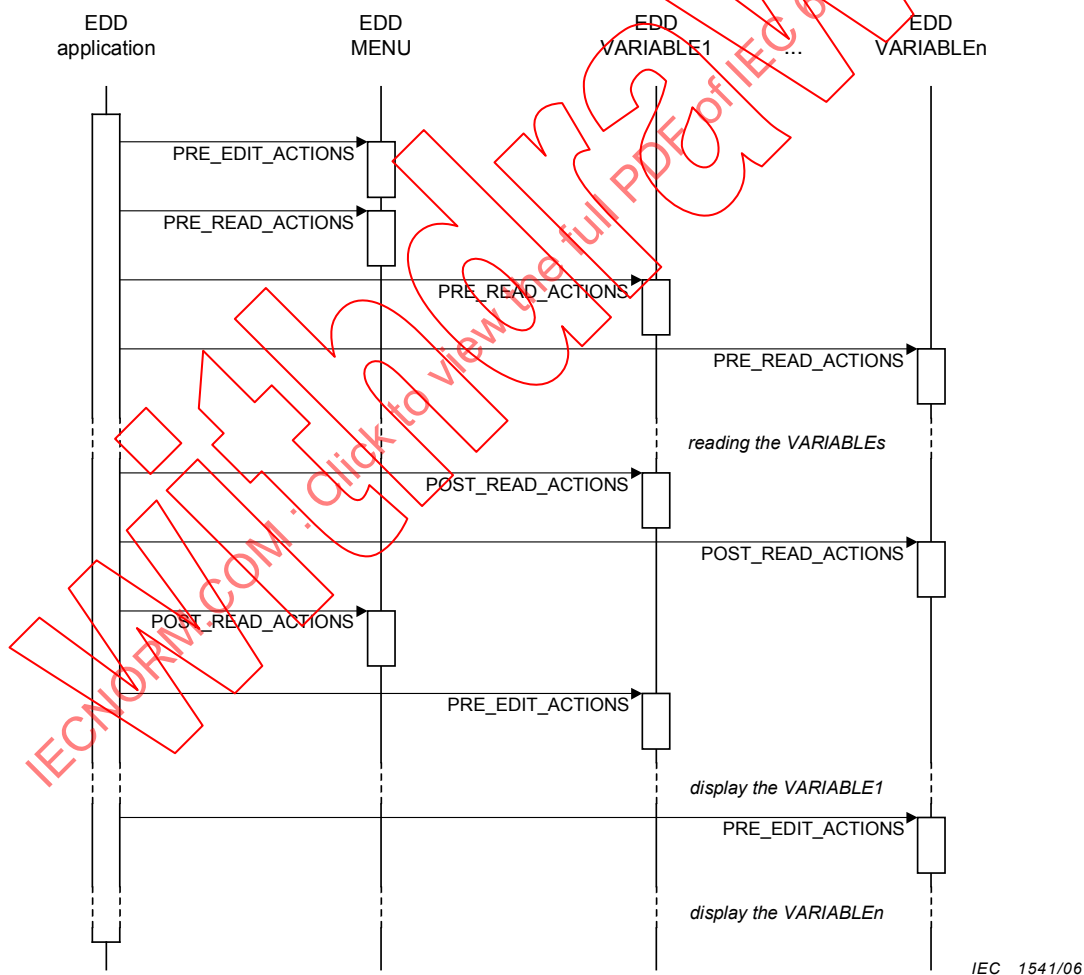
Anglais	Français
EDD application	Application EDD

Figure 30 – Action réalisée après que toutes les entrées de VARIABLE du MENU ont été acceptées (ACCESS OFFLINE)



En cas d'accès en ligne, les valeurs modifiées de VARIABLE doivent être écrites dans l'appareil avant d'exécuter la METHOD. Après l'exécution de la METHOD, les valeurs sont relues dans l'appareil, puis affichées.

Figure 31 – Exécution de la méthode

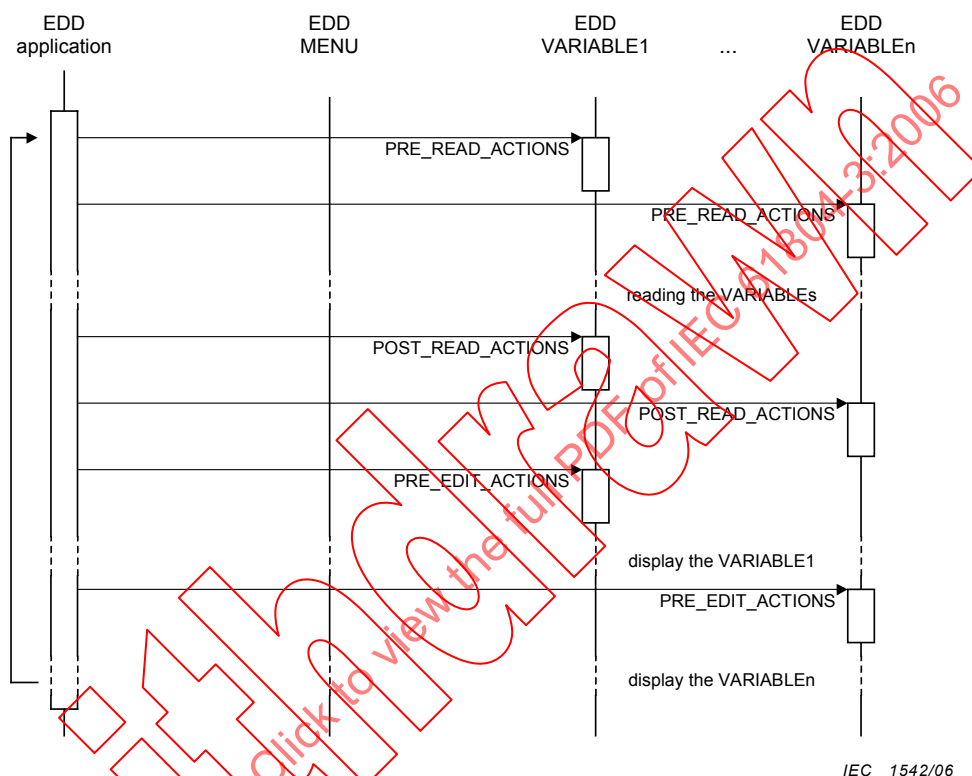


Légende

Anglais	Français
EDD application	Application EDD
reading the VARIABLES	lecture des VARIABLE
display the VARIABLE1	afficher la VARIABLE1
display the VARIABLEn	afficher la VARIABLEn

- a Pour les VARIABLES lues en ligne à partir d'un appareil, l'application EDD exécute les COMMAND appropriées. Les VARIABLES peuvent inclure des actions de lecture préalable/d'écriture qui doivent précéder la COMMAND appropriée. Les actions de lecture préalable/d'écriture des VARIABLES doivent être exécutées suite à la COMMAND appropriée.
- b Chaque VARIABLE ou autre entité est présentée à l'utilisateur à l'issue des PRE_EDIT_ACTIONS.
- c L'ordre des PRE_READ_ACTIONS et POST_READ_ACTIONS des différentes VARIABLE peut être aléatoire; elles doivent être exécutées avant MENU POST_READ_ACTIONS.

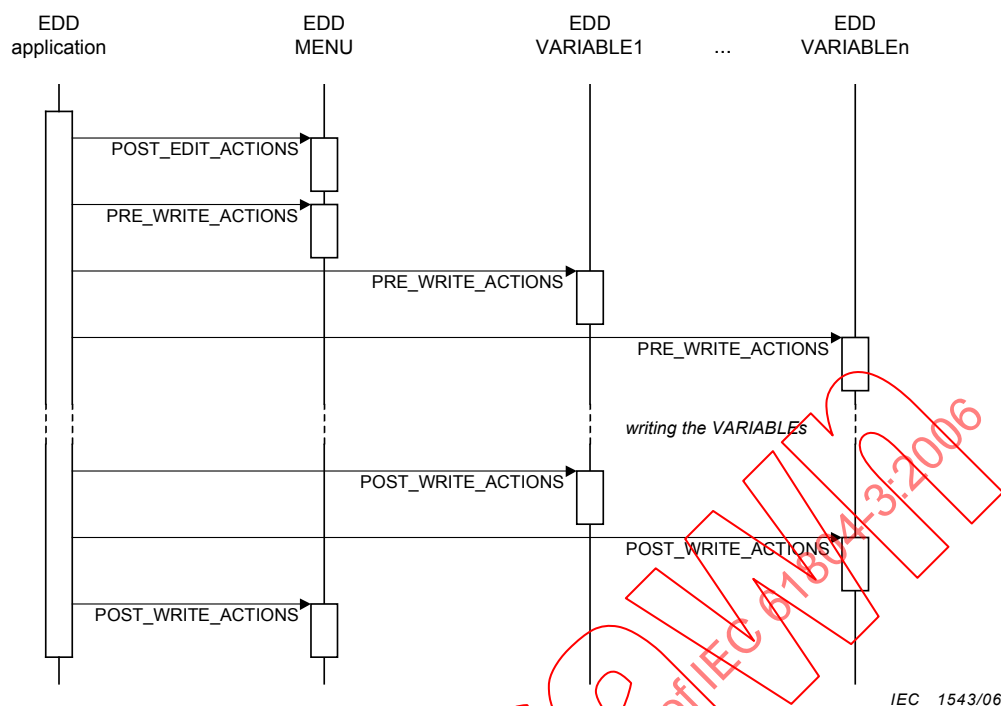
Figure 32 – Activation de MENU (ACCESS ONLINE)



Légende

Anglais	Français
EDD application	Application EDD
reading the VARIABLES	lecture des VARIABLES
display the VARIABLE1	afficher la VARIABLE1
display the VARIABLEn	afficher la VARIABLEn

Figure 33 – Lecture cyclique des VARIABLES dynamiques (ACCESS ONLINE)

**Légende**

Anglais	Français
EDD application	Application EDD
writing the VARIABLES	écriture des VARIABLES

Les METHODS sont exécutées après que toutes les entrées de VARIABLE sont acceptées. Seules les valeurs de VARIABLE modifiées sont écrites dans l'appareil. Après que les modifications ont été écrites, les VARIABLES doivent être lues pour assurer la cohérence.

Figure 34 – Action réalisée après que toutes les entrées de VARIABLE du MENU sont acceptées (ACCESS ONLINE)

7.19 METHOD

7.19.1 Structure générale

Objet

Une METHOD est utilisée pour définir un sous-programme qui est exécuté dans l'application EDD.

NOTE Une METHOD permet de spécifier les vérifications de cohérence, les algorithmes de conversion entre les variables EDD, les acquittements utilisateur ou les interactions de l'appareil spécifique.

Structure lexicale

METHOD identifier [DEFINITION, LABEL, ACCESS, CLASS, HELP, TYPE, VALIDITY]

Les attributs de METHOD sont spécifiés au Tableau 117.

Tableau 117 – Attributs METHOD

Usage	Attribut	Description
m	DEFINITION	Élément lexical (voir 7.30.1)
m	LABEL	Élément lexical (voir 7.30.3)
o	ACCESS	Élément lexical (voir 7.19.2.1)
o	CLASS	Élément lexical (voir 7.19.2.2)

Usage	Attribut	Description
o	HELP	Élément lexical (voir 7.30.2)
o	TYPE	Élément lexical (voir 7.19.2.3)
o	VALIDITY	Élément lexical (voir 7.19.2.3)

7.19.2 Attributs spécifiques

7.19.2.1 ACCESS

Objet

L'attribut ACCESS spécifie si les valeurs de variable stockées dans l'appareil ou l'ensemble de données hors ligne sont utilisées dans la DEFINITION de METHOD. Si ACCESS n'est pas défini, la valeur par défaut est ONLINE.

Structure lexicale

ACCESS (OFFLINE, ONLINE) <exp>

Les attributs d'ACCESS sont spécifiés au Tableau 118.

Tableau 118 – Attributs d'ACCESS

Usage	Attribut	Description
s	OFFLINE	Spécifie que les valeurs des variables utilisées dans la définition de METHOD sont lues ou écrites depuis/vers l'ensemble de données hors connexion
s	ONLINE	Spécifie que les valeurs des variables utilisées dans la définition de METHOD sont effectivement lues depuis l'appareil. Les changements de valeurs de VARIABLE sont écrits avec des Builtin.

7.19.2.2 CLASS

Objet

L'attribut CLASS (classe) spécifie l'effet d'une méthode sur un dispositif de terrain. Cet attribut est destiné à être utilisé par l'application EDD pour implémenter des niveaux d'autorisation et définit comment des METHOD sont présentées. Les classes de METHOD sont identiques aux classes de VARIABLE.

Structure lexicale

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

Les attributs de CLASS sont spécifiés Tableau 139.

7.19.2.3 TYPE

Objet

L'attribut TYPE spécifie le type de la valeur retournée.

Structure lexicale

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

Les attributs de VALIDITY sont spécifiés au Tableau 120.

Tableau 119 – Attributs de TYPE

Usage	Attribut	Description
s	char	Spécifie que la METHOD renvoie une valeur de type char
s	unsigned char	Spécifie que la METHOD renvoie une valeur de type char non signé
s	int	Spécifie que la METHOD renvoie une valeur de type entier
s	unsigned int	Spécifie que la METHOD renvoie une valeur de type entier non signé
s	long	Spécifie que la METHOD renvoie une valeur de type long
s	unsigned long	Spécifie que la METHOD renvoie une valeur de type long non signé
s	float	Spécifie que la METHOD renvoie une valeur de type virgule flottante
s	double	Spécifie que la METHOD renvoie une valeur de type double
s	DD_STRING	Spécifie que la METHOD renvoie une valeur de type DD_STRING

7.19.2.4 VALIDITY

Objet

L'attribut VALIDITY spécifie si la METHOD est exécutée ou pas. Une METHOD sans attribut VALIDITY est toujours valide.

NOTE VALIDITY est en principe exprimé à l'aide d'une expression conditionnelle (voir 7.32).

Structure lexicale

VALIDITY [(FALSE, TRUE)<exp>]

Les attributs de VALIDITY sont spécifiés au Tableau 120.

Tableau 120 – Attributs de VALIDITY

Usage	Attribut	Description
s	FALSE	Spécifie que la METHOD ne doit pas être exécutée
s	TRUE	Spécifie que la METHOD doit être exécutée

7.20 PROGRAM

7.20.1 Structure générale

Objet

PROGRAM permet la commande à distance pour l'exécution d'un programme.

EXAMPLE Un appareil peut télécharger un PROGRAM dans un domaine d'un autre appareil en utilisant un service de téléchargement, puis commander à distance le PROGRAM en émettant une demande de service. Il convient que des PROGRAM puissent être créés, supprimés, démarrés, arrêtés, etc.

Structure lexicale

PROGRAM identifier [ARGUMENT, RESPONSE_CODES]

Les attributs de PROGRAM sont spécifiés au Tableau 121.

Tableau 121 – Attribut de PROGRAM

Usage	Attribut	Description
o	ARGUMENT	Élément lexical (voir 7.20.2.1)
o	RESPONSE_CODES	Élément lexical (voir 7.30.5)

7.20.2 Attributs spécifiques

7.20.2.1 ARGUMENT

Objet

Les ARGUMENTs peuvent être envoyés au programme pendant le démarrage et la reprise des opérations. Les arguments de PROGRAM sont décrits par l'attribut ARGUMENT. Une chaîne d'octet contenant les valeurs de tous les arguments est envoyée à l'objet qui invoque le programme lorsqu'il est démarré ou repris par l'application EDD.

Structure lexicale

ARGUMENT [(data-item)<exp>]+

L'attribut d'ARGUMENT est spécifié au Tableau 122.

Tableau 122 – Attribut ARGUMENT

Usage	Attribut	Description
m	data-item	Constante entière non signée ou VARIABLE. Si une donnée est une constante entière non signée, la valeur de la constante apparaît à cette position dans le champ de données. Les données constantes occupent deux octets du champ de données et par conséquent, doivent être dans l'intervalle de 0 à 65535 inclus. Si une donnée est une VARIABLE, la valeur de la variable apparaît à cette position dans le champ de données. Si le champ de données du service PROGRAM est vide, les arguments peuvent être omis

7.20.2.2 Généralités

PROGRAM ne comporte qu'un seul attribut spécifique.

7.21 RECORD

Objet

Un RECORD est un groupe logique de VARIABLE. Chaque élément dans le RECORD est une référence à une VARIABLE et peut être de différents types de données.

NOTE Un RECORD peut être transféré sous la forme d'un objet de communication.

Structure lexicale

RECORD identifier [MEMBERS, LABEL, HELP, RESPONSE_CODES]

Les attributs de RECORD sont spécifiés au Tableau 123

Tableau 123 – Attributs de RECORD

Usage	Attribut	Description
m	MEMBERS	Élément lexical (voir 7.30.4)
m	LABEL	Élément lexical (voir 7.30.3)
o	HELP	Élément lexical (voir 7.30.2)
o	RESPONSE_CODES	Élément lexical (voir 7.30.5)

7.22 REFERENCE_ARRAY

7.22.1 Structure générale

Objet

Un REFERENCE_ARRAY est un ensemble d'éléments du type d'élément spécifié (par exemple, VARIABLE ou MENU).

NOTE Les REFERENCE_ARRAY ne sont pas liés aux matrices de communication (type d'élément "VALUE_ARRAY"). Les matrices de communication sont des matrices de valeurs.

Structure lexicale

REFERENCE_ARRAY identifier [item-type, ELEMENTS, HELP, LABEL]

Les attributs de REFERENCE_ARRAY sont spécifiés au Tableau 124.

Tableau 124 – Attribut de REFERENCE_ARRAY

Usage	Attribut	Description
m	item-type	Spécifie le type d'élément figurant dans l'attribut ELEMENTS. Doit être l'un des éléments de base répertoriés au Tableau 42.
m	ELEMENTS	Élément lexical (voir 7.22.2.1)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)

7.22.2 Attributs spécifiques

7.22.2.1 ELEMENTS

Objet

L'attribut ELEMENTS (éléments) spécifie une liste d'éléments. Il définit pour chaque élément la référence d'élément, son index associé, sa description et une aide facultatives. Les attributs de description et d'aide, s'ils sont utilisés, remplacent les spécifications correspondantes dans l'EDD elle-même.

Structure lexicale

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

Les attributs d'ELEMENTS sont spécifiés au Tableau 125.

Tableau 125 – Attributs d'ELEMENTS

Usage	Attribut	Description
m	integer	Spécifie le numéro par lequel l'élément peut être référencé. Chaque numéro doit être défini de façon unique dans une construction REFERENCE_ARRAY
m	reference	Référence à une instance de data-item
o	description	Fournit une brève description de l'élément
o	help	spécifie le texte d'aide pour l'élément

7.22.2.2 Généralités

REFERENCE_ARRAY ne comporte qu'un seul attribut spécifique.

7.23 Relations

7.23.1 Structure générale

Objet

Les relations spécifient les relations entre des instances des VARIABLES. Les types de relations suivants sont définis:

- REFRESH (voir 7.23.2)
- UNIT (voir 7.23.3)
- WRITE_AS_ONE (voir 7.23.4)

7.23.2 REFRESH

Objet

Une relation REFRESH est utilisée pendant l'accès en ligne et spécifie un ensemble de VARIABLE qu'il convient de réactualiser (lire depuis l'appareil) chaque fois qu'un ensemble spécifié est modifié.

NOTE Une VARIABLE peut avoir une relation REFRESH avec elle-même, ce qui signifie que la variable doit être lue après écriture.

Structure lexicale

REFRESH *identifieur*, [(cause-parameter)<exp>]+, [(effected-parameter)<exp>]+

Les attributs de REFRESH sont spécifiés au Tableau 126.

Tableau 126 – Attributs de REFRESH

Usage	Attribut	Description
m	cause-parameter	Spécifie une liste des instances de VARIABLE
m	effected-parameter	Spécifie une liste d'instances de VARIABLE, qui doivent être lues, si l'un des paramètres de cause a été modifié. Une VARIABLE de cette liste de paramètres effectués peut également se trouver dans la liste de paramètres de cause d'une autre relation REFRESH

7.23.3 UNIT

Objet

Une relation UNIT (unité) spécifie une référence à une VARIABLE contenant un code d'unité et une liste de VARIABLE dépendantes. Lorsque la variable contenant le code d'unité est modifiée, la liste de VARIABLE/AXIS réalisés utilisant ce code d'unité doit être réactualisée. Lorsqu'un VARIABLE/AXIS réalisé est affiché, la valeur de son code d'unité doit également être affichée.

Structure lexicale

UNIT *identifieur*, [cause-parameter)<exp>], [(effected-parameter)<exp>]+

Les attributs d'UNIT sont spécifiés au Tableau 127.

Tableau 127 – Attributs d'UNIT

Usage	Attribut	Description
m	cause-parameter	Référence à une instance de VARIABLE qui contient le code d'unité
m	effected-parameter	Spécifie une liste d'instances de VARIABLE/AXIS utilisant ce code d'unité

7.23.4 WRITE_AS_ONE

Objet

Une relation WRITE_AS_ONE spécifie une liste de VARIABLES qui doit être modifiée en tant que groupe par l'application EDD. Cette relation est utilisée lorsqu'un dispositif requiert que des VARIABLES spécifiques soient analysées et modifiées simultanément pour un fonctionnement correct.

NOTE Cette relation ne signifie pas nécessairement que les VARIABLES sont écrites simultanément vers l'appareil. Toutes les VARIABLES envoyées en même temps à l'appareil ne font pas nécessairement partie d'une relation WRITE_AS_ONE. Il revient à l'application EDD de déterminer dans quelle mesure les mises à jour des paramètres de cette relation sont transmises à l'appareil.

Structure lexicale

WRITE_AS_ONE *identifieur*, [(*reference*)<exp>]+

L'attribut de WRITE_AS_ONE est spécifié au Tableau 128.

Tableau 128 – Attribut de WRITE_AS_ONE

Usage	Attribut	Description
m	reference	Spécifie une liste d'instances de VARIABLES, VALUE_ARRAY et RECORD qui doivent être modifiées dans un groupe avec les autres membres de la relation WRITE_AS_ONE

7.24 RESPONSE_CODES

Objet

Les RESPONSE_CODES (codes de réponse) spécifient les valeurs qu'un dispositif peut retourner en tant qu'information d'erreur. Chaque VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST, PROGRAM, COMMAND ou DOMAIN peut avoir son propre ensemble de RESPONSE_CODES.

Structure lexicale

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

Les attributs de RESPONSE_CODES sont spécifiés au Tableau 129.

Tableau 129 – Attributs de RESPONSE_CODES

Usage	Attribut	Description
m	description	Spécifie une chaîne qui s'affiche lorsque le RESPONSE_CODE est renvoyé par l'appareil
m	integer	Spécifie la valeur renvoyée de l'appareil pour identifier le RESPONSE_CODE
s	DATA ENTRY ERROR	Le service de couche application a été rejeté parce que les données envoyées ne sont pas valides
s	DATA ENTRY WARNING	Le service de couche application a été accepté et traité avec une version sensiblement modifiée des données envoyées
s	MISC ERROR	Le service de couche application a été rejeté
s	MISC WARNING	Le service de couche application a été accepté et traité comme spécifié, et il existe des informations supplémentaires, sans rapport avec le service de couche application, susceptibles d'intéresser l'utilisateur
s	MODE ERROR	Le service de couche application a été rejeté parce que l'appareil de terrain était dans un mode dans lequel le service de couche application ne pouvait pas être exécuté

Usage	Attribut	Description
s	PROCESS ERROR	Le service de couche application a été rejeté parce qu'un processus appliqué à l'appareil de terrain n'était pas valide
s	SUCCESS	Le service de couche application a été accepté et traité tel que spécifié
o	help	Spécifie le texte d'aide pour l'élément

7.25 SOURCE

7.25.1 Structure générale

Objet

Une SOURCE décrit une source de valeurs de données pour un CHART (diagramme)

Structure lexicale

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

Les attributs de SOURCE sont spécifiés au Tableau 130.

Tableau 130 – Attributs de SOURCE

Usage	Attribut	Description
m	MEMBERS	Élément lexical (voir 7.30.4)
o	EMPHASIS	Élément lexical (voir 7.25.2.1)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)
o	LINE_COLOR	Élément lexical (voir 7.25.2.2)
o	LINE_TYPE	Élément lexical (voir 7.25.2.3)
o	Y_AXIS	Élément lexical (voir 7.25.2.4)
o	EXIT_ACTIONS	Élément lexical (voir 7.25.2.5)
o	INIT_ACTIONS	Élément lexical (voir 7.25.2.6)
o	REFRESH_ACTIONS	Élément lexical (voir 7.25.2.7)

7.25.2 Attributs spécifiques

7.25.2.1 EMPHASIS

Objet

L'attribut EMPHASIS spécifie s'il convient ou pas de mettre en évidence la SOURCE lorsqu'elle est affichée. La manière de mettre en évidence une SOURCE est définie par l'application EDD. Par défaut, il convient de ne pas mettre en évidence une SOURCE sans attribut EMPHASIS.

Structure lexicale

EMPHASIS (FALSE, TRUE) <exp>

Les attributs d'EMPHASIS sont spécifiés au Tableau 131.

Tableau 131 – Attribut d'EMPHASIS

Usage	Attribut	Description
s	FALSE	Spécifie qu'il convient de ne pas mettre en évidence la SOURCE
s	TRUE	Spécifie qu'il convient de mettre en évidence la SOURCE

7.25.2.2 LINE_COLOR**Objet**

L'attribut LINE_COLOR spécifie la couleur qu'il convient d'utiliser pour afficher la SOURCE. Par défaut, une SOURCE sans attribut LINE_COLOR s'affiche en utilisant une couleur par défaut définie par l'application EDD.

Structure lexicale

LINE_COLOR (integer, reference)<exp>

Les couleurs sont spécifiées par l'intermédiaire d'une valeur RVB, comme défini dans Cascading Style Sheets, niveau 2.

Les attributs de LINE_COLOR sont spécifiés au Tableau 132.

Tableau 132 – Attribut de LINE_COLOR

Usage	Attribut	Description
s	integer	Couleur à utiliser pour afficher la SOURCE
s	reference	Référence à une instance de VARIABLE contenant la couleur à utiliser pour afficher la SOURCE.

7.25.2.3 LINE_TYPE**Objet**

L'attribut LINE_TYPE spécifie le type de données contenues dans la SOURCE. Il convient d'afficher chaque SOURCE avec le même attribut LINE_TYPE de la même façon (c'est-à-dire, couleur de trait, motif de trait, épaisseur de trait, etc.). Par défaut, une SOURCE sans attribut LINE_TYPE utilise le style de trait DATA.

Structure lexicale

LINE_TYPE ([DATA, n]*, LOW_LOW_LIMIT, LOW_LIMIT, HIGH_LIMIT, HIGH_HIGH_LIMIT)<exp>

Les attributs de LINE_TYPE sont spécifiés au Tableau 133.

Tableau 133 – Attributs de LINE_TYPE

Usage	Attribut	Description
s	DATA	Spécifie que la SOURCE contient des données
c	n	Entier croissant commençant à 1, qui identifie des types de trait DATA distincts si plusieurs sont présents
s	LOW_LOW_LIMIT	Spécifie que la SOURCE contient une limite très basse
s	LOW_LIMIT	Spécifie que la SOURCE contient une limite basse
s	HIGH_LIMIT	Spécifie que la SOURCE contient une limite élevée
s	HIGH_HIGH_LIMIT	Spécifie que la SOURCE contient une limite très élevée

7.25.2.4 Y_AXIS

Objet

L'attribut Y_AXIS spécifie le mode d'affichage de l'axe Y de la SOURCE. Chaque SOURCE sur un CHART partage un axe X commun qui est basé sur le temps, mais chaque SOURCE sur un CHART à son propre Y_AXIS.

Structure lexicale

Y_AXIS (reference) <exp>

L'attribut de Y_AXIS est spécifié au Tableau 134.

Tableau 134 – Attribut d'Y_AXIS

Usage	Attribut	Description
m	reference	Référence à une instance d'AXIS

7.25.2.5 EXIT_ACTIONS

Objet

Les EXIT_ACTIONS sont exécutées lorsque le CHART contenant la SOURCE est fermé.

Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHODS restantes ne doivent pas être exécutées.

Structure lexicale

EXIT_ACTIONS [(reference) <exp>]+, DEFINITION

Les attributs d'EXIT_ACTIONS sont spécifiés au Tableau 167.

7.25.2.6 INIT_ACTIONS

Objet

Les INIT_ACTIONS sont exécutées avant que la valeur de SOURCE soit initialement affichée par un CHART.

Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHODS restantes ne doivent pas être exécutées. De plus, l'application EDD ne doit pas représenter graphiquement la valeur de SOURCE si une méthode se termine pour une raison imprévue.

Structure lexicale

INIT_ACTIONS [(reference) <exp>]+, DEFINITION

Les attributs d'INIT_ACTIONS sont spécifiés au Tableau 167.

7.25.2.7 REFRESH_ACTIONS

Les REFRESH_ACTIONS sont exécutées avant l'affichage de la valeur sur le CHART dans lequel la SOURCE est affichée.

Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHODS restantes ne doivent pas être exécutées.

Structure lexicale

REFRESH_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de REFRESH_ACTIONS sont spécifiés au Tableau 167.

7.26 VALUE_ARRAY

7.26.1 Structure générale

Objet

Un VALUE_ARRAY (matrice de valeurs) est un groupe logique de valeurs. Chaque élément d'un VALUE_ARRAY doit être du même type de données et doit correspondre à une construction EDDL. Un élément est référencé depuis un autre emplacement dans l'EDD via l'identifiant et l'index d'élément de VALUE_ARRAY.

NOTE Du point de vue de la communication, les éléments individuels du VALUE_ARRAY ne sont pas traités comme des variables individuelles, mais simplement comme une valeur groupée.

Structure lexicale

VALUE_ARRAY identifieur [LABEL, NUMBER_OF_ELEMENTS, TYPE, HELP, RESPONSE_CODES]

Les attributs de VALUE_ARRAY sont spécifiés au Tableau 135.

Tableau 135 – Attributs de VALUE_ARRAY

Usage	Attribut	Description
m	LABEL	Élément lexical (voir 7.30.3)
m	NUMBER_OF_ELEMENTS	Élément lexical (voir 7.26.2.1)
m	TYPE	Élément lexical (voir 7.26.2.2)
o	HELP	Élément lexical (voir 7.30.2)
o	RESPONSE_CODES	Élément lexical (voir 7.30.5)

7.26.2 Attributs spécifiques

7.26.2.1 NUMBER_OF_ELEMENTS

Objet

L'attribut NUMBER_OF_ELEMENTS spécifie le nombre d'éléments dans le VALUE_ARRAY (entier supérieur à zéro).

Structure lexicale

NUMBER_OF_ELEMENTS integer, expression)

L'attribut de NUMBER_OF_ELEMENTS est spécifié au Tableau 136.

Tableau 136 – Attributs de NUMBER_OF_ELEMENTS

Usage	Attribut	Description
s	integer	Spécifie le nombre d'éléments d'un VALUE_ARRAY
s	expression	Expression qui doit être évaluée à un entier non négatif compris dans l'intervalle d'index de l'attribut NUMBER_OF_ELEMENTS (pour la description des expressions, voir 7.34.7).

7.26.2.2 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 VALUE_ARRAY.

Structure lexicale

TYPE reference

L'attribut de TYPE est spécifié au Tableau 137.

Tableau 137 – Attribut de TYPE

Usage	Attribut	Description
m	reference	Référence à une instance de construction de base d'EDD

7.27 VARIABLE

7.27.1 Structure générale

Objet

Une VARIABLE décrit des paramètres contenus dans un appareil ou dans l'application EDD.

Structure lexicale

VARIABLE identifier, [CLASS, LABEL, TYPE, HELP, CONSTANT_UNIT, DEFAULT_VALUE, PRE_EDIT_ACTIONS, POST_EDIT_ACTIONS, PRE_READ_ACTIONS, POST_READ_ACTIONS, PRE_WRITE_ACTIONS, POST_WRITE_ACTIONS, REFRESH_ACTIONS, READ_TIMEOUT, WRITE_TIMEOUT, RESPONSE_CODE, HANDLING, VALIDITY]

Les attributs de VARIABLE sont spécifiés au Tableau 138.

Tableau 138 – Attributs de VARIABLE

Usage	Attribut	Description
m	CLASS	Élément lexical (voir 7.27.2.1)
m	LABEL	Élément lexical (voir 7.30.3)
m	TYPE	Élément lexical (voir 7.27.2.2)
o	CONSTANT_UNIT	Élément lexical (voir 7.27.2.3)
o	DEFAULT_VALUE	Élément lexical (voir X.X.X)
o	HANDLING	Élément lexical (voir 7.27.2.4)
o	HELP	Élément lexical (voir 7.30.2)
o	INITIAL_VALUE	Élément lexical (voir X.X.X)
o	POST_EDIT_ACTIONS	Élément lexical (voir 7.27.2.6)
o	POST_READ_ACTIONS	Élément lexical (voir 7.27.2.8)
o	POST_WRITE_ACTIONS	Élément lexical (voir 7.27.2.9)
o	PRE_EDIT_ACTIONS	Élément lexical (voir 7.27.2.10)
o	PRE_READ_ACTIONS	Élément lexical (voir 7.27.2.11)
o	PRE_WRITE_ACTIONS	Élément lexical (voir 7.27.2.12)
o	REFRESH_ACTIONS	Élément lexical (voir 7.27.2.13)
o	READ_TIMEOUT	Élément lexical (voir 7.27.2.14)

Usage	Attribut	Description
o	RESPONSE_CODE	Élément lexical (voir 7.30.5)
o	STYLE	Élément lexical (voir 7.27.2.15)
o	VALIDITY	Élément lexical (voir 7.27.2.16)
o	WRITE_TIMEOUT	Élément lexical (voir 7.27.2.13)

7.27.2 Attributs spécifiques

7.27.2.1 CLASS

Objet

L'attribut CLASS spécifie comment la VARIABLE est utilisée par l'appareil et l'application EDD à des fins d'organisation et d'affichage.

Les combinaisons de classes ne sont pas toutes autorisées. Les restrictions sont définies dans les tables de profils (voir Annexe D).

NOTE La définition de CLASS repose sur un modèle dans lequel un appareil est composé d'une pluralité de blocs fonctionnels différents. Chaque bloc a une entrée et produit une sortie. Des blocs peuvent être combinés de différentes façons pour produire différentes fonctionnalités.

Structure lexicale

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

Les attributs de CLASS sont spécifiés au Tableau 139.

Tableau 139 – Attributs de CLASS

Usage	Attribut	Description
s	ALARM	Spécifie que cette VARIABLE contient des limites d'alarme
s	ANALOG_INPUT	Spécifie que cette VARIABLE fait partie d'un bloc d'entrée analogique
s	ANALOG_OUTPUT	Spécifie que cette VARIABLE fait partie d'un bloc de sortie analogique
s	COMPUTATION	Spécifie que cette VARIABLE fait partie d'un bloc de calcul
s	CONTAINED	Spécifie que cette VARIABLE représente les caractéristiques physiques de l'appareil
s	CORRECTION	Spécifie que cette VARIABLE fait partie du bloc de correction
s	DEVICE	Spécifie que cette VARIABLE représente les caractéristiques physiques de l'appareil
s	DIAGNOSTIC	Spécifie que cette VARIABLE indique le statut de l'appareil
s	DIGITAL_INPUT	Spécifie que cette VARIABLE fait partie d'un bloc d'entrée numérique
s	DIGITAL_OUTPUT	Spécifie que cette VARIABLE fait partie d'un bloc de sortie numérique
s	DISCRETE_INPUT	Spécifie que cette VARIABLE fait partie d'un bloc d'entrée discret
s	DISCRETE_OUTPUT	Spécifie que cette VARIABLE fait partie d'un bloc de sortie discret
s	DYNAMIC	Spécifie que cette VARIABLE est modifiée par l'appareil sans stimulus provenant du réseau
s	FREQUENCY_INPUT	Spécifie que cette VARIABLE fait partie d'un bloc d'entrée de fréquence
s	FREQUENCY_OUTPUT	Spécifie que cette VARIABLE fait partie d'un bloc de sortie de fréquence
s	HART	Spécifie que cette VARIABLE fait partie du bloc HART qui caractérise l'interface HART. Il n'existe qu'un seul bloc HART

Usage	Attribut	Description
s	INPUT	Spécifie que cette VARIABLE fait partie d'un bloc d'entrée. Un bloc d'entrée est un type particulier de bloc de calcul qui effectue des conversions d'unité, une mise à l'échelle et un amortissement. Le paramètre des paramètres de bloc d'entrée peut être déterminé par la sortie d'un autre bloc
s	LOCAL	Spécifie que cette VARIABLE est utilisée localement par l'application EDD. Les VARIABLES locales ne sont pas stockées dans un appareil, mais elles peuvent être envoyées à un appareil
s	LOCAL_DISPLAY	Spécifie que cette VARIABLE fait partie du bloc d'affichage local. Un bloc d'affichage local contient les VARIABLES associées à l'interface locale (clavier, affichage, etc.) de l'appareil
s	OPERATE	Spécifie que cette VARIABLE est utilisée pour contrôler le fonctionnement d'un bloc
s	OUTPUT	Spécifie que cette VARIABLE fait partie du bloc de sortie. Les valeurs de VARIABLES de sortie peuvent être accessibles par un autre bloc d'entrée
s	SERVICE	Spécifie que cette VARIABLE est utilisée lors de la conduite de maintenance de routine
s	TUNE	Spécifie que cette VARIABLE est utilisée pour régler l'algorithme d'un bloc

7.27.2.2 TYPE

7.27.2.2.1 Structure générale

Objet

L'attribut TYPE décrit le format des valeurs de VARIABLE. Si TYPE est spécifié sans DEFAULT_VALUE et INITIAL_VALUE, la colonne "Initial value" du Tableau 140 présente les valeurs initiales.

Structure lexicale

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

Les attributs de TYPE sont spécifiés au Tableau 140.

Tableau 140 – Attributs de TYPE

Usage	Attribut	Description	Valeur initiale
s	DOUBLE	Type arithmétique (voir 7.27.2.2.2.1).	1
s	FLOAT	Type arithmétique (voir 7.27.2.2.2.1).	1
s	INTEGER	Type arithmétique (voir 7.27.2.2.2.1).	1
s	UNSIGNED_INTEGER	Type arithmétique (voir 7.27.2.2.2.1).	1
s	DATE	Types de date/heure (voir 7.27.2.2.3).	
s	DATE_AND_TIME	Types de date/heure (voir 7.27.2.2.3).	
s	DURATION	Types de date/heure (voir 7.27.2.2.3).	
s	TIME	Types de date/heure (voir 7.27.2.2.3).	00:00:00
s	TIME_VALUE	Types de date/heure (voir 7.27.2.2.3).	
s	BIT_ENUMERATED	Type d'énumération (voir 7.27.2.2.3.1).	0
s	ENUMERATED	Type d'énumération (voir 7.27.2.2.3.2).	Valeur de la première
s	INDEX	Type d'index (voir 7.27.2.2.3.3).	
s	OBJECT_REFERENCE	Références d'objet (voir 7.27.2.2.4).	
s	ASCII	Type de chaîne (voir 7.27.2.2.5).	

Usage	Attribut	Description	Valeur initiale
s	BITSTRING	Type de chaîne (voir 7.27.2.2.5).	
s	EUC	Type de chaîne (voir 7.27.2.2.5).	
s	OCTET	Type de chaîne (voir 7.27.2.2.5).	
s	PACKED_ASCII	Type de chaîne (voir 7.27.2.2.5).	
s	PASSWORD	Type de chaîne (voir 7.27.2.2.5).	
s	VISIBLE	Type de chaîne (voir 7.27.2.2.5).	

7.27.2.2.2 Attributs spécifiques

7.27.2.2.2.1 DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Objet

Les VARIABLES de TYPE FLOAT et DOUBLE sont des nombres à virgule flottante au format de base simple précision et au format de base double précision, comme défini dans l'IEEE 754, respectivement. Les VARIABLES de TYPE INTEGER et UNSIGNED_INTEGER sont respectivement des nombres entiers signés et non signés.

Ces types sont appelés collectivement types arithmétiques.

Structure lexicale — DOUBLE

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

Structure lexicale — FLOAT

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

Structure lexicale — INTEGER

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

Structure lexicale — UNSIGNED_INTEGER

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

Les attributs des types arithmétiques sont spécifiés au Tableau 141.

Tableau 141 – Attributs de DOUBLE, FLOAT, INTEGER, UNSIGNED_INTEGER

Usage	Attribut	Description
o	DEFAULT_VALUE	Objet La DEFAULT_VALUE spécifie le réglage par défaut pour la variable. La DEFAULT_VALUE est disponible pour tous les types EDDL. Si aucune INITIAL_VALUE n'est définie, la DEFAULT_VALUE est utilisée en tant qu'INITIAL_VALUE. La DEFAULT_VALUE peut également être générée par une expression (voir 7.35). Structure lexicale DEFAULT_VALUE (double)<exp> DEFAULT_VALUE (float)<exp> DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT_VALUE (expression)<exp>
o	description	Brève description de l'élément
o	DISPLAY_FORMAT	Objet Un DISPLAY_FORMAT spécifie l'affichage de la valeur de la VARIABLE. Les chaînes DISPLAY_FORMAT sont des spécificateurs de conversion des fonctions ANSI C (voir l'ISO/CEI 9899) printf et scanf, respectivement Structure lexicale DISPLAY_FORMAT (string)<exp>
o	EDIT_FORMAT	Objet Un attribut EDIT_FORMAT spécifie le format pour l'édition. Les chaînes EDIT_FORMAT sont des spécificateurs de conversion des fonctions ANSI C (voir l'ISO/CEI 9899) printf et scanf, respectivement Structure lexicale EDIT_FORMAT (string)<exp>
o	help	Spécifie le texte d'aide pour l'élément
o	INITIAL_VALUE	Objet Lorsqu'un appareil est instancié pour la première fois, l'INITIAL_VALUE d'une VARIABLE est affichée. Il est fréquent que la DEFAULT_VALUE soit conditionnée et dépende d'autres paramètres. Dans ce cas, l'INITIAL_VALUE est utilisée pour reconstituer les paramètres par défaut. INITIAL_VALUE est disponible pour tous les types et ne doit faire l'objet d'aucune condition Structure lexicale INITIAL_VALUE (double) INITIAL_VALUE (float) INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)
o	MAX_VALUE	Objet MAX_VALUE spécifie la limite supérieure des valeurs qui doivent être attribuées à une VARIABLE. Si la VARIABLE est CLASS DYNAMIC, il convient que l'appareil attribue une valeur à la VARIABLE comprise dans l'intervalle spécifié par ses valeurs minimales et maximales. Une VARIABLE peut avoir plusieurs valeurs maximales (pour spécifier plusieurs intervalles de validité, par exemple). Lorsqu'il y a plusieurs valeurs maximales, un entier supplémentaire est utilisé pour numéroté chaque MAX_VALUE. Chaque MAX_VALUE doit avoir une MIN_VALUE équivalente. L'intervalle spécifié avec une paire de MIN_VALUE et MAX_VALUE peut ne pas chevaucher un autre intervalle spécifié. Si une MIN_VALUE/MAX_VALUE n'a pas de MAX_VALUE/MIN_VALUE équivalente, la limite supérieure/inférieure est illimitée et peut ne pas être spécifiée. La MAX_VALUE peut également être construite par une expression (voir 7.35) Structure lexicale MAX_VALUE (double, integer)<exp> MAX_VALUE (float, integer)<exp> MAX_VALUE (integer, integer)<exp> MAX_VALUE (unsigned_integer, integer)<exp> MAX_VALUE (expression)<exp>

Usage	Attribut	Description
c	m	Entier croissant commençant à 1, qui identifie la MAX_VALUE si il y en plusieurs
o	MIN_VALUE	Objet MIN_VALUE spécifie la limite inférieure des valeurs qui doivent être attribuées à une VARIABLE. Si la VARIABLE est CLASS DYNAMIC, il convient que l'appareil attribue une valeur à la VARIABLE comprise dans l'intervalle spécifié par ses valeurs minimale et maximale Une VARIABLE peut avoir plusieurs valeurs minimales (pour spécifier plusieurs intervalles de validité, par exemple). S'il existe plusieurs valeurs minimales, un entier supplémentaire est utilisé pour numéroté chaque MIN_VALUE Chaque MIN_VALUE doit avoir une MAX_VALUE équivalente. L'intervalle spécifié avec une paire de MIN_VALUE et MAX_VALUE ne peut pas chevaucher un autre intervalle spécifié. Si une MIN_VALUE/MAX_VALUE n'a pas de MAX_VALUE/MIN_VALUE équivalente, la limite supérieure/inférieure est illimitée et peut ne pas être spécifiée. La MIN_VALUE peut également être construite par une expression (voir 7.35) Structure lexicale MIN_VALUE (double, integer)<exp> MIN_VALUE (float, integer)<exp> MIN_VALUE (integer, integer)<exp> MIN_VALUE (unsigned_integer, integer)<exp> MIN_VALUE (expression)<exp>
c	n	Entier croissant commençant à 1, qui identifie la MIN_VALUE si il y en plusieurs
o	SCALING_FACTOR	Objet SCALING_FACTOR indique que la valeur affichée de la VARIABLE n'est pas celle renvoyée par un appareil. La valeur affichée est la valeur retournée par un dispositif multipliée par un facteur. Par conséquent, le processeur EDDL doit multiplier la valeur de la variable retournée par un dispositif avec son SCALING_FACTOR avant qu'elle soit affichée (ou avant qu'elle soit utilisée autrement). Le SCALING_FACTOR ne doit pas être égal à zéro. Le SCALING_FACTOR peut également être construit par une expression (voir 7.35) Structure lexicale SCALING_FACTOR (double)<exp> SCALING_FACTOR (expression)<exp>
o	size	Spécifie la taille du type de données en octets. La taille est une constante entière supérieure à zéro et n'a pas de limite supérieure. Cette valeur est facultative. La valeur par défaut est 1
o	value	Constante qui spécifie la valeur. Le type dépend du type de données spécifié avec l'attribut TYPE qui est facultatif, mais s'il est défini, une description d'attribut est requise. Les valeurs égales ne sont pas admises

7.27.2.2.3 DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE

Objet

Les VARIABLES de type de données DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE sont utilisées pour spécifier des dates:

- 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 composée d'un temps en ms et d'un compteur de jours facultatif;
- le type de données TIME est constitué d'une heure et d'une date facultative;
- le type de données TIME_VALUE est utilisé pour représenter la date et l'heure avec la précision requise pour la synchronisation de l'horloge d'application.

Structure lexicale — DATE

DATE spécifie que la VARIABLE est un entier codé sur trois octets. Le codage des octets est spécifié en D.5.2.

DATE DEFAULT_VALUE, INITIAL_VALUE

Structure lexicale – DATE_AND_TIME

DATE_AND_TIME spécifie que la VARIABLE est un entier codé sur sept octets. Le codage des octets est spécifié en D.5.3.

DATE_AND_TIME DEFAULT_VALUE, INITIAL_VALUE

Structure lexicale — DURATION

DURATION spécifie que la VARIABLE est un entier de 4 à 6 octets. Les quatre premiers bits doivent avoir la valeur zéro. Les bits 0 à 27 représentent l'heure en ms. Le nombre de jours facultatif est une valeur binaire de 16 bits. Le codage des octets est spécifié en D.5.4.

DURATION DEFAULT_VALUE, INITIAL_VALUE

Structure lexicale — TIME

TIME spécifie que la VARIABLE est un entier codé sur six octets. L'heure est exprimée en ms à partir de minuit. A minuit, le compteur recommence à zéro.

La date est établie en jours à partir du 01/01/1984. Le premier jour du mois de janvier 1984, le comptage de date commence par une valeur nulle. Le codage des octets est spécifié en D.5.5.

TIME DEFAULT_VALUE, INITIAL_VALUE

Structure lexicale – TIME_VALUE

TIME_VALUE spécifie que la VARIABLE est un entier codé sur huit octets. C'est un nombre à décimale fixe, qui est le temps exprimé en multiples de 1/32 ms. Le codage des octets est spécifié en D.5.6.

TIME_VALUE DEFAULT_VALUE, INITIAL_VALUE

Les attributs des types DATE, DATE_AND_TIME, DURATION, TIME, TIME_VALUE sont spécifiés au Tableau 141.

7.27.2.2.3.1 BIT_ENUMERATED

Objet

Les variables de type de données BIT_ENUMERATED sont des constantes entières non signées qui identifient un bit de la valeur de variable avec sa position.

Structure lexicale

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

Les attributs du type BIT_ENUMERATED sont spécifiés au Tableau 142.

Tableau 142 – Attributs de BIT_ENUMERATED

Usage	Attribut	Description
m	description	Voir Tableau 141
m	value	Entier non signé qui spécifie une position de bit. Si la valeur de la variable affecte aux positions de bit spécifiées un 1 logique, la description associée est affichée. Les valeurs égales ne sont pas admises
o	actions	Spécifie les actions exécutées lorsque le bit est défini
o	function	Spécifie la classe fonctionnelle du bit. La classe fonctionnelle d'un bit est la même que la CLASS d'une VARIABLE. Si aucune fonction n'est spécifiée, la valeur de la classe de fonction est égale par défaut à la classe de la variable. Par conséquent, si tous les bits ont la même fonction, la CLASS de la VARIABLE doit être spécifiée
o	help	Voir Tableau 141
o	size	Voir Tableau 141
o	status-class	Spécifie la signification du bit. Un bit de statut peut appartenir à plusieurs status-class (voir Tableau 143). Si la VARIABLE n'est pas un octet de statut, les bits ne comportent pas de classe de statut
o	DEFAULT_VALUE	Objet La DEFAULT_VALUE spécifie le réglage par défaut pour la variable. La DEFAULT_VALUE est disponible pour tous les types EDDL. Si aucune INITIAL_VALUE n'est définie, la DEFAULT_VALUE est utilisée en tant qu'INITIAL_VALUE. La DEFAULT_VALUE peut également être construite par une expression (voir 7.35) Structure lexicale DEFAULT_VALUE (integer)<exp> DEFAULT_VALUE (unsigned_integer)<exp> DEFAULT (VALUE (expression)<exp>
o	INITIAL_VALUE	Objet Lorsqu'un appareil est instancié pour la première fois, l'INITIAL_VALUE d'une VARIABLE est affichée. Il est fréquent que la DEFAULT_VALUE soit conditionnée et dépende d'autres paramètres. Dans ce cas, l'INITIAL_VALUE est utilisée pour reconstituer les paramètres par défaut. INITIAL_VALUE est disponible pour tous les types et ne doit faire l'objet d'aucune condition Structure lexicale INITIAL_VALUE (integer) INITIAL_VALUE (unsigned_integer)

Le Tableau 143 présente les status-class pour une VARIABLE de type BIT_ENUMERATED.

Tableau 143 – Attributs de status-class

Usage	Attribut	Description
m	DATA	Indique une configuration de données non valide
m	HARDWARE	Indique une défaillance du matériel
m	MISC	Indique des conditions diverses
m	MODE	Indique que l'appareil est dans un mode particulier
m	PROCESS	Indique un problème avec le processus connecté à l'appareil de terrain
m	SOFTWARE	Indique une défaillance du logiciel
m	CORRECTABLE	Indique que le bit peut être effacé par l'application EDD ou le processus connecté
m	SELF_CORRECTING	Indique que le bit est réinitialisé sans intervention supplémentaire
m	UNCORRECTABLE	Indique que le bit n'est ni auto-corrigeable, ni corrigeable
m	EVENT	Indique un événement unique
m	STATE	Indique que l'appareil est dans un statut particulier.
m	COMM_ERROR	Indique une défaillance de la communication dans l'autre appareil
o	IGNORE_IN_HANDHELD	Indique qu'il convient d'ignorer un bit dans les appareils de communication portables

Usage	Attribut	Description
o	IGNORE_IN_TEMPORARY_MASTER	Indique qu'il convient d'ignorer un bit dans des appareils maîtres temporaires
m	MORE	Indique que des informations de statut supplémentaires sont disponibles en provenance de l'appareil
o	ALL	Indique l'impact sur toutes les sorties (voir Tableau 144)
o	AO	Indique l'impact sur l'une des sorties analogiques (voir Tableau 144)
m	BAD	Indique que la sortie n'est pas fiable et qu'il convient de ne pas l'utiliser pour la commande (voir Tableau 144)
o	DV	Indique un impact sur l'une des variables dynamiques (voir Tableau 144)
o	TV	Indique l'impact sur l'une des variables de l'émetteur (voir Tableau 144)
m	DETAIL	Indique que le bit est répété ailleurs dans un bit de classe SUMMARY
m	SUMMARY	Indique que le bit est une combinaison logique d'autres bits de classe DETAIL

Les status-class ALL, AO, DV et TV donnent des informations de statut supplémentaires sur la sortie d'un dispositif.

Structure lexicale — ALL

ALL [AUTO_BAD, AUTO_GOOD, MANUAL_BAD, MANUAL_GOOD]

Structure lexicale — AO

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

Structure lexicale — DV

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

Structure lexicale — TV

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

Les attributs sont spécifiés au Tableau 144.

Tableau 144 – Attributs ALL, AO, DV, TV

Usage	Attribut	Description
s	AUTO_BAD	Spécifie que la valeur provient d'un processus d'application et qu'elle n'est pas valide
s	AUTO_GOOD	Spécifie que la valeur provient d'un processus d'application et qu'elle n'est pas valide
m	integer	Spécifie que la valeur de VARIABLE provient de l'appareil
s	MANUAL_BAD	La valeur ne provient pas d'un processus d'application et n'est pas valide
s	MANUAL_GOOD	La valeur ne provient pas d'un processus d'application et n'est pas valide

7.27.2.2.3.2 ENUMERATED

Objet

Une VARIABLE de type ENUMERATED est un entier non signé. Une liste associée de triplets (entier, description, aide) spécifie les valeurs possibles de la variable ainsi qu'une description et une aide facultative.

Structure lexicale

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

Les attributs du type ENUMERATED sont spécifiés au Tableau 145.

Tableau 145 – Attributs de types énumérés

Usage	Attribut	Description
m	description	(voir Tableau 141)
m	value	Constante qui spécifie la valeur VARIABLE. Le type est un entier non signé. La liste d'énumération est facultative mais si elle est définie, une valeur et une description sont requises. Les valeurs égales ne sont pas admises.
o	DEFAULT_VALUE	(voir Tableau 141)
o	help	(voir Tableau 141)
o	INITIAL_VALUE	(voir Tableau 141)
o	size	(voir Tableau 141)

7.27.2.2.3.3 INDEX

Objet

Une VARIABLE de TYPE INDEX est un entier non signé qui est utilisé pour référencer les éléments d'un REFERENCE_ARRAY (voir 7.22). L'attribut entier des ELEMENTS (voir 7.22.2.1) spécifie les valeurs autorisées de la VARIABLE.

NOTE Lorsqu'une VARIABLE INDEX est présentée à l'utilisateur, le texte associé aux index de la matrice doit être affiché, mais pas la valeur numérique de la VARIABLE.

Structure lexicale

INDEX (reference, size)<exp>

Les attributs du type d'index sont spécifiés au Tableau 146.

Tableau 146 – Attributs de TYPE INDEX

Usage	Attribut	Description
o	size	Spécifie la taille de la VARIABLE en octets. Cette valeur est une constante entière supérieure à zéro et n'a pas de limite supérieure. La valeur par défaut est 1. La REFERENCE_ARRAY doit contenir uniquement des éléments qui ne dépassent pas la taille d'index de la variable.
m	reference	Référence à une instance de REFERENCE_ARRAY

7.27.2.2.4 Types de référence d'objet

Objet

Les VARIABLES du type OBJECT_REFERENCE permettent d'accéder aux valeurs d'autres dispositifs.

Structure lexicale

OBJECT_REFERENCE DEFAULT_REFERENCE

Les attributs des types OBJECT_REFERENCE sont spécifiés au Tableau 147.

Tableau 147 – Attribut du type de référence d'objet

Usage	Attribut	Description
o	DEFAULT_REFERENCE	Spécifie que la VARIABLE est utilisée pour accéder aux valeurs d'autres appareils. Si aucune DEFAULT_REFERENCE n'est définie, la VARIABLE est initialisée avec SELF (voir Tableau 148)

Les attributs de DEFAULT-REFERENCE sont spécifiés au Tableau 148.

Tableau 148 – Attributs de DEFAULT_REFERENCE

Usage	Attribut	Description
s	CHILD	Navigue d'un niveau hiérarchique vers le bas
s	FIRST	Navigue jusqu'au premier objet de la hiérarchie
s	LAST	Navigue jusqu'au dernier objet de la hiérarchie
s	NEXT	Navigue jusqu'à l'objet suivant de la même hiérarchie
s	PARENT	Navigue dans une hiérarchie de bas en haut
s	PREV	Navigue vers l'objet précédent de la même hiérarchie
s	reference	Référence à un appareil projeté dans le réseau
s	SELF	Retourne vers lui-même dans sa propre hiérarchie
s	NULL	Référence d'objet vide

7.27.2.2.5 ASCII, BIT_STRING, EUC, PACKED_ASCII, OCTET, PASSWORD, VISIBLE

Objet

Les variables de chaîne comprennent les types suivants.

- ASCII – utilisé pour spécifier une séquence de caractères du jeu de caractères ISO Latin-1.
- BITSTRING – Les variables de type de données BIT_ENUMERATED sont des constantes de chaîne qui identifient la position d'un caractère de la valeur de VARIABLE par rapport à sa position.
- EUC (codé d'unité étendu) – est le code interne pour des caractères spécifiques (par exemple, Chine: EUC-CN, Taiwan: EUC-TW). EUC est utilisé pour gérer les langues asiatiques (ISO/CEI 10646-1).
- Le processeur EDDL doit prendre en charge un schéma de codage dans lequel des caractères sont codés en utilisant un nombre variable d'octets. Typiquement, certains octets signalent le début d'un caractère et combien d'octets additionnels sont utilisés pour coder le caractère. Des jeux de caractères avec un grand nombre d'éléments sont souvent stockés en utilisant un schéma d'encapsulation.
- PACKED_ASCII – est un sous-ensemble d'ASCII obtenu en enlevant les deux bits les plus significatifs de chaque caractère ASCII (voir D.5.7).
- PASSWORD – est un type chaîne utilisé pour spécifier des chaînes des mots de passe. À l'exception du mode de présentation de la variable à l'utilisateur, les types de chaîne de mot de passe et ASCII sont identiques.
- VISIBLE – Cette chaîne est une séquence ordonnée de caractères du jeu de caractères de l'ISO/CEI 10646-1 et l'ISO 2375.
- OCTET – est utilisé pour spécifier une séquence de données binaires non formatées dont la définition est déterminée par l'implémentation de l'appareil.

Structure lexicale — EUC

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

Structure lexicale — ASCII

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

Structure lexicale — PACKED_ASCII

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

Structure lexicale — PASSWORD

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

Structure lexicale — BITSTRING

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

Structure lexicale — VISIBLE

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

Structure lexicale — OCTET

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

Les attributs des types de chaîne sont spécifiés au Tableau 149.

Tableau 149 – Attributs de types de chaîne

Usage	Attribut	Description
o	size	(voir Tableau 141)
o	DEFAULT_VALUE	(voir Tableau 141)
o	INITIAL_VALUE	(voir Tableau 141)
o	string	Constante de chaîne. La liste d'énumération est facultative mais si elle est définie, une chaîne et une description sont requises. Des chaînes égales ne sont pas autorisées.
o	description	(voir Tableau 141)
o	help	(voir Tableau 141)

7.27.2.3 CONSTANT_UNIT**Objet**

L'attribut CONSTANT_UNIT (unité constante) est utilisé, si une VARIABLE a un code d'unité qui ne change jamais. CONSTANT_UNIT est spécifiée comme étant une chaîne qui est affichée avec la valeur. Une VARIABLE sans unité constante ne comporte pas d'unités associées à celle-ci ou bien les unités ne sont pas constantes.

Structure lexicale

```
CONSTANT_UNIT (string)<exp>
```

L'attribut de CONSTANT_UNIT est spécifié au Tableau 150.

Tableau 150 – Attribut CONSTANT_UNIT

Usage	Attribut	Description
m	string	Spécifie la chaîne de code d'unités à afficher

7.27.2.4 DEFAULT_VALUE**Objet**

L'attribut DEFAULT_VALUE (valeur par défaut) spécifie le réglage par défaut pour la variable. Si aucune INITIAL_VALUE n'est définie, la DEFAULT_VALUE est utilisée en tant qu'INITIAL_VALUE.

Structure lexicale

DEFAULT_VALUE (expression)<exp>

NOTE Le type de l'expression doit être cohérent avec le type de la VARIABLE.

NOTE La DEFAULT_VALUE peut également être spécifiée via l'attribut TYPE de la VARIABLE.

L'attribut de DEFAULT_VALUE est spécifié au Tableau 150.

Tableau 151 – Attribut de DEFAULT_VALUE

Usage	Attribut	Description
m	expression	Spécifie la valeur par défaut de la variable

7.27.2.5 HANDLING**Objet**

L'attribut HANDLING spécifie les opérations qui peuvent être effectuées sur la VARIABLE. Par défaut, une VARIABLE sans attribut HANDLING peut être lue et écrite.

Structure lexicale

HANDLING (READ, READ_WRITE, WRITE)<exp>

Les attributs de HANDLING sont spécifiés au Tableau 152.

Tableau 152 – Attribut de HANDLING

Usage	Attribut	Description
s	READ	Spécifie que la valeur d'une VARIABLE peut être lue
s	READ_WRITE	Spécifie que la valeur d'une VARIABLE peut être lue et écrite
s	WRITE	Spécifie que la valeur d'une VARIABLE peut être écrite

7.27.2.6 INITIAL_VALUE**Objet**

Lorsqu'un appareil est instancié pour la première fois, l'INITIAL_VALUE d'une VARIABLE est affichée. Il est fréquent que la DEFAULT_VALUE soit conditionnée et dépende d'autres paramètres. Dans ce cas, l'INITIAL_VALUE est utilisée pour reconstituer les paramètres par défaut. L'INITIAL_VALUE est disponible pour tous les types et ne doit pas être conditionnée.

Structure lexicale

Expression d'INITIAL_VALUE

NOTE 1 Le type de l'expression doit être cohérent avec le type de la VARIABLE.

NOTE 2 L'INITIAL_VALUE peut également être spécifiée via l'attribut TYPE de la VARIABLE.

L'attribut d'INITIAL_VALUE est spécifié au Tableau 150.

Tableau 153 – Attribut d'INITIAL_VALUE

Usage	Attribut	Description
m	expression	Spécifie la valeur initiale de la variable

7.27.2.7 POST_EDIT_ACTIONS

Objet

L'attribut POST_EDIT_ACTIONS (actions post-édition) spécifie les METHODS qui doivent être exécutées après que l'utilisateur ait terminé d'éditer la VARIABLE. Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHODS suivantes ne sont pas exécutées.

Structure lexicale

POST_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de POST_EDIT_ACTIONS sont spécifiés au Tableau 154.

Tableau 154 – Attributs de POST_EDIT_ACTIONS, PRE_EDIT_ACTIONS, POST_READ_ACTIONS, PRE_READ_ACTIONS, POST_WRITE_ACTIONS, PRE_WRITE_ACTIONS, REFRESH_ACTIONS

Usage	Attribut	Description
o	reference	Référence à une instance de METHOD
o	DEFINITION	Élément lexical (voir 7.30.1)

7.27.2.8 POST_READ_ACTIONS

Objet

L'attribut POST_READ_ACTIONS spécifie les METHODS qui doivent être exécutées après que la variable ait été lue depuis l'appareil. Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHODS suivantes ne sont pas exécutées.

Structure lexicale

POST_READ_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de POST_READ_ACTIONS sont spécifiés au Tableau 154.

7.27.2.9 POST_WRITE_ACTIONS

Objet

L'attribut POST_WRITE_ACTIONS spécifie les METHODS qui doivent être exécutées après que la variable ait été écrite vers l'appareil. Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHODS suivantes ne sont pas exécutées.

Structure lexicale

POST_WRITE_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de POST_WRITE_ACTIONS sont spécifiés au Tableau 154.

7.27.2.10 PRE_EDIT_ACTIONS

Objet

L'attribut PRE_EDIT_ACTIONS spécifie les METHODS qui doivent être exécutées immédiatement après l'affichage de la VARIABLE. Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHODS suivantes ne sont pas exécutées.

Structure lexicale

PRE_EDIT_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de PRE_EDIT_ACTIONS sont spécifiés au Tableau 154.

7.27.2.11 PRE_READ_ACTIONS

Objet

L'attribut PRE_READ_ACTIONS spécifie les METHODS qui doivent être exécutées avant que la variable soit lue. Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHODS suivantes ne sont pas exécutées.

Structure lexicale

PRE_READ_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de PRE_READ_ACTIONS sont spécifiés au Tableau 154.

7.27.2.12 PRE_WRITE_ACTIONS

Objet

L'attribut PRE_WRITE_ACTIONS spécifie les METHODS qui doivent être exécutées avant que la variable soit écrite dans l'appareil. Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHODS suivantes ne sont pas exécutées.

Structure lexicale

PRE_WRITE_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de PRE_WRITE_ACTIONS sont spécifiés au Tableau 154.

7.27.2.13 REFRESH_ACTIONS

Objet

L'attribut REFRESH_ACTIONS spécifie les METHODS qui doivent être exécutées lorsque la variable est affichée ou réactualisée. Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. S'il existe une METHOD pour une raison imprévue, les METHODS suivantes ne sont pas exécutées.

Structure lexicale

REFRESH_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs de REFRESH_ACTIONS sont spécifiés au Tableau 154.

7.27.2.14 READ_TIMEOUT, WRITE_TIMEOUT

Objet

Un READ_TIMEOUT spécifie la durée, en ms, pendant laquelle l'application EDD doit attendre la VARIABLE renvoyée. De même, un WRITE_TIMEOUT spécifie la durée, en ms, pendant laquelle une application EDD doit attendre la confirmation d'écriture de la variable dans l'appareil.

Par exemple, il convient qu'un WRITE_TIMEOUT indique la durée nécessaire pour qu'un dispositif stocke la valeur d'une variable.

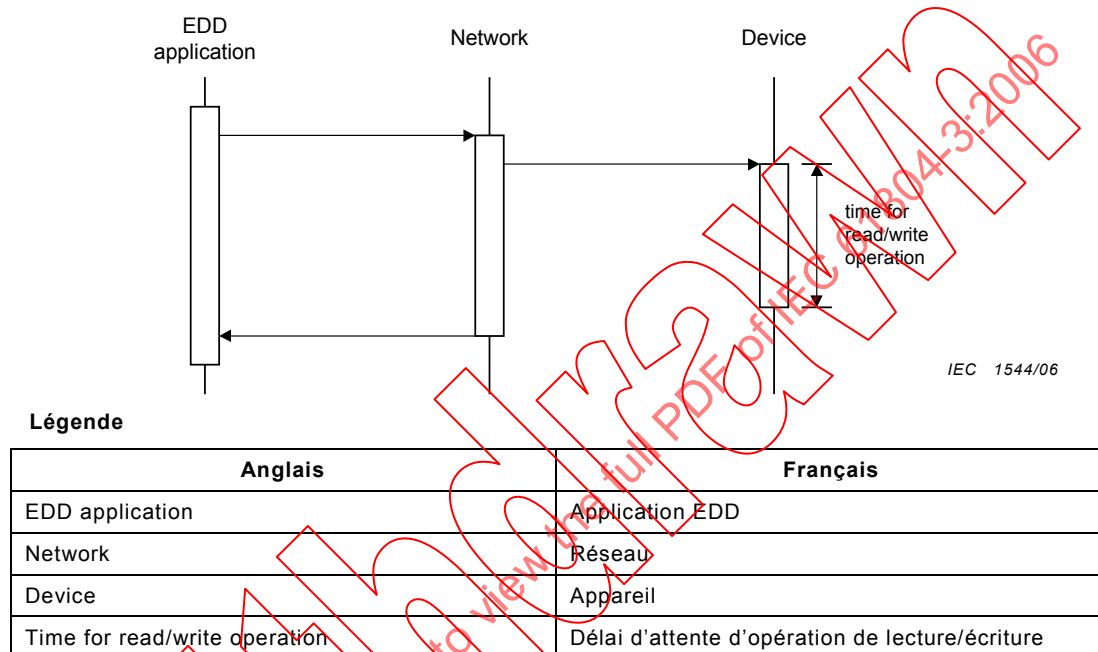


Figure 35 – Délai d'attente d'opération de lecture/écriture

Structure lexicale

READ_TIMEOUT (integer) <exp>
WRITE_TIMEOUT (integer) <exp>

L'attribut de READ_TIMEOUT et WRITE_TIMEOUT est spécifié au Tableau 155.

Tableau 155 – Attributs de READ/WRITE_TIMEOUT

Usage	Attribut	Description
m	integer	Spécifie le temps, en ms, pendant lequel l'application EDD attend qu'un appareil termine sa mise à jour demandée

7.27.2.15 STYLE

Objet

L'attribut STYLE spécifie le mode d'affichage d'une VARIABLE. C'est une information additionnelle qui est individuellement interprétée par l'application EDDL.

Structure lexicale

STYLE (string) <exp>

L'attribut de STYLE est spécifié au Tableau 156.

Tableau 156 – Attribut de STYLE

Usage	Attribut	Description
m	string	Spécifie une chaîne utilisée pour l'affichage

7.27.2.16 VALIDITY**Objet**

L'attribut VALIDITY spécifie si une VARIABLE est ou n'est pas valide. Une VARIABLE n'est pas valide uniquement si elle n'existe pas. Si sa valeur est simplement erronée, VARIABLE est toujours valide. Une VARIABLE sans attribut VALIDITY est toujours valide.

NOTE 1 Il convient d'exprimer VALIDITY en utilisant une instruction conditionnelle (IF, IF-ELSE, SELECT).

NOTE 2 Si une VARIABLE n'est pas valide, il convient de ne pas l'afficher.

Structure lexicale

VALIDITY [(FALSE, TRUE) <exp>]

Les attributs de VALIDITY sont spécifiés au Tableau 157.

Tableau 157 – Attributs de VALIDITY

Usage	Attribut	Description
s	FALSE	Spécifie que l'instance de VARIABLE n'est pas valide
s	TRUE	Spécifie que l'instance de VARIABLE est valide

7.28 VARIABLE_LIST**Objet**

Une VARIABLE_LIST est un groupe d'objets de communication d'EDD (VARIABLE, VALUE_ARRAY ou RECORDS). Les VARIABLE_LISTS sont utilisés pour grouper des objets afin de faciliter l'exécution des applications.

NOTE Une VARIABLE_LIST peut être transférée sous la forme d'un objet de communication.

Structure lexicale

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

Les attributs sont spécifiés au Tableau 158.

Tableau 158 – Attributs de VARIABLE_LIST

Usage	Attribut	Description
m	MEMBERS	Élément lexical (voir 7.30.4)
o	HELP	Élément lexical (voir 7.30.2)
o	LABEL	Élément lexical (voir 7.30.3)
o	RESPONSE_CODES	Élément lexical (voir 7.30.5)

7.29 WAVEFORM

7.29.1 Structure générale

Objet

Une WAVEFORM (forme d'onde) décrit un ensemble de données qui peut être affiché par un GRAPH (graphique).

Structure lexicale

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

Les attributs de WAVEFORM sont spécifiés au Tableau 159.

Tableau 159 – Attributs de WAVEFORM

Usage	Attribut	Description
m	TYPE	Élément lexical (voir 7.29.2.1)
o	EMPHASIS	Élément lexical (voir 7.29.2.2)
o	EXIT_ACTIONS	Élément lexical (voir 7.29.2.3)
o	HANDLING	Élément lexical (voir 7.29.2.4)
o	HELP	Élément lexical (voir 7.30.2)
o	INIT_ACTIONS	Élément lexical (voir 7.29.2.5)
o	KEY_POINTS	Élément lexical (voir 7.29.2.6)
o	LABEL	Élément lexical (voir 7.30.3)
o	LINE_COLOR	Élément lexical (voir 7.29.2.7)
o	LINE_TYPE	Élément lexical (voir 7.29.2.8)
o	REFRESH_ACTIONS	Élément lexical (voir 7.29.2.9)
o	Y_AXIS	Élément lexical (voir 7.29.2.10)

7.29.2 Attributs spécifiques

7.29.2.1 TYPE

7.29.2.1.1 Structure générale

Objet

L'attribut TYPE spécifie le type de données contenues dans la WAVEFORM.

Structure lexicale

TYPE [XY, YT, HORIZONTAL, VERTICAL]

Les attributs de TYPE sont spécifiés au Tableau 160.

Tableau 160 – Attribut de TYPE

Usage	Attribut	Description
s	XY	Liste de points (x, y) (voir 7.29.2.1.2.1)
s	YT	Liste de points (y, t) (voir 7.29.2.1.2.2)
s	HORIZONTAL	Liste de traits horizontaux (voir 7.29.2.1.2.3)
s	VERTICAL	Liste de traits verticaux (voir 7.29.2.1.2.4)

7.29.2.1.2 Attributs spécifiques

7.29.2.1.2.1 XY

Objet

L'attribut XY spécifie une WAVEFORM qui contient une liste de points (x,y).

Structure lexicale

XY [X_VALUES, Y_VALUES, NUMBER_OF_POINTS]

Les attributs de XY sont spécifiés au Tableau 161.

Tableau 161 – Attribut de XY

Usage	Attribut	Description
m	X_VALUES	Objet L'attribut X_VALUES spécifie la coordonnée X de chaque point dans la WAVEFORM. Chaque coordonnée X spécifiée dans X_VALUES doit correspondre à une coordonnée Y spécifiée dans Y_VALUES. Structure lexicale X_VALUES [reference, double, float, integer, unsigned_integer]+ Les attributs de X_VALUES sont spécifiés au Tableau 169.
m	Y_VALUES	Objet L'attribut Y_VALUES spécifie la coordonnée Y de chaque point dans la WAVEFORM. A chaque coordonnée Y spécifiée dans Y_VALUES doit correspondre une coordonnée X spécifiée dans X_VALUES. Structure lexicale Y_VALUES [(reference, double, float, integer, unsigned_integer)+ Les attributs de Y_VALUES sont spécifiés au Tableau 169.
o	NUMBER_OF_POINTS	Objet L'attribut NUMBER_OF_POINTS spécifie le nombre de points valides dans X_VALUES et Y_VALUES. Par défaut, le nombre de points dans une WAVEFORM sans attribut NUMBER_OF_POINTS est égal à la taille de X_VALUES et Y_VALUES Structure lexicale NUMBER_OF_POINTS (integer)<exp>.

7.29.2.1.2.2 YT

Objet

L'attribut YT spécifie une WAVEFORM qui contient une liste de points qui sont définis par une coordonnée X initiale, un incrément X entre des points successifs et une liste de valeurs Y.

Structure lexicale

YT [X_INITIAL, X_INCREMENT, Y_VALUES, NUMBER_OF_POINTS]

Les attributs de YT sont spécifiés au Tableau 162.

Tableau 162 – Attribut de YT

Usage	Attribut	Description
m	X_INITIAL	Objet L'attribut X_INITIAL spécifie la coordonnée X du premier point dans la WAVEFORM. Structure lexicale X_INITIAL (reference)<exp> X_INITIAL (double)<exp> X_INITIAL (float)<exp> X_INITIAL (integer)<exp>

Usage	Attribut	Description
		X_INITIAL (unsigned_integer)<exp> La référence est une référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY ou un élément d'une instance de REFERENCE_ARRAY.
m	X_INCREMENT	Objet L'attribut X_INCREMENT spécifie la différence entre les coordonnées X de points adjacents dans la WAVEFORM. Structure lexicale X_INCREMENT (reference)<exp> X_INCREMENT (double)<exp> X_INCREMENT (float)<exp> X_INCREMENT (integer)<exp> X_INCREMENT (unsigned_integer)<exp> La référence est une référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY ou un élément d'une instance de REFERENCE_ARRAY.
m	Y_VALUES	Objet L'attribut Y_VALUES spécifie la coordonnée Y de chaque point dans la WAVEFORM. Structure lexicale Y_VALUES [(reference, double, float, integer, unsigned_integer)<exp>]+ Les attributs de Y_VALUES sont spécifiés au Tableau 169.
o	NUMBER_OF_POINTS	Objet L'attribut NUMBER_OF_POINTS spécifie le nombre de points valides dans X_VALUES et Y_VALUES. Par défaut, le nombre de points dans une WAVEFORM sans attribut NUMBER_OF_POINTS est égal à la taille de X_VALUES et Y_VALUES. Structure lexicale NUMBER_OF_POINTS (integer)<exp>.

7.29.2.1.2.3 HORIZONTAL

Objet

L'attribut HORIZONTAL spécifie une WAVEFORM qui contient des lignes horizontales.

Structure lexicale

HORIZONTAL Y_VALUES

L'attribut de HORIZONTAL est spécifié au Tableau 163.

Tableau 163 – Attribut de HORIZONTAL

Usage	Attribut	Description
m	Y_VALUES	Objet L'attribut Y_VALUES spécifie la coordonnée Y de chaque ligne horizontale dans la WAVEFORM. Structure lexicale Y_VALUES [reference, double, float, integer, unsigned_integer]+ Les attributs de Y_VALUES sont spécifiés au Tableau 169.

7.29.2.1.2.4 VERTICAL

Objet

L'attribut VERTICAL spécifie une WAVEFORM qui contient des lignes verticales.

Structure lexicale

VERTICAL X_VALUES

Les attributs de VERTICAL sont spécifiés au Tableau 164.

Tableau 164 – Attribut de VERTICAL

Usage	Attribut	Description
m	X_VALUES	Objet L'attribut X_VALUES spécifie la coordonnée X de chaque ligne verticale dans la WAVEFORM. Structure lexicale X_VALUES [reference, double, float, integer, unsigned_integer]+ Les attributs de X_VALUES sont spécifiés au Tableau 169.

7.29.2.2 EMPHASIS**Objet**

L'attribut EMPHASIS spécifie s'il convient ou pas de mettre en évidence la WAVEFORM lorsqu'elle est affichée. La manière de mettre en évidence une WAVEFORM est définie par l'application EDD. Par défaut, il convient de ne pas mettre en évidence une WAVEFORM sans attribut EMPHASIS.

Structure lexicale

EMPHASIS (FALSE, TRUE) <exp>

Les attributs d'EMPHASIS sont spécifiés au Tableau 165.

Tableau 165 – Attribut d'EMPHASIS

Usage	Attribut	Description
s	FALSE	Spécifie qu'il convient de ne pas mettre en évidence la WAVEFORM
s	TRUE	Spécifie qu'il convient de mettre en évidence la WAVEFORM

7.29.2.3 EXIT_ACTIONS**Objet**

Les EXIT_ACTIONS sont exécutées lorsque le GRAPH contenant la WAVEFORM est fermé.

Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHODS restantes ne doivent pas être exécutées.

Structure lexicale

EXIT_ACTIONS [(reference) <exp>]+, DEFINITION

Les attributs d'EXIT_ACTIONS sont spécifiés au Tableau 167.

7.29.2.4 HANDLING**Objet**

L'attribut HANDLING spécifie les opérations qui peuvent être effectuées sur la WAVEFORM. Par défaut, une WAVEFORM sans attribut HANDLING peut être lue et écrite.

Structure lexicale

HANDLING (READ, READ_WRITE, WRITE) <exp>

Les attributs de HANDLING sont spécifiés au Tableau 166.

Tableau 166 – Attribut de HANDLING

Usage	Attribut	Description
s	READ	Spécifie que la WAVEFORM peut uniquement être lue
s	READ_WRITE	Spécifie que la WAVEFORM peut être lue et écrite
s	WRITE	Spécifie que la WAVEFORM peut être écrite

7.29.2.5 INIT_ACTIONS**Objet**

Les INIT_ACTIONS sont exécutées avant que la WAVEFORM soit initialement affichée par un GRAPH. Ces méthodes sont généralement utilisées pour charger la WAVEFORM avec les données provenant d'un dispositif ou d'un FILE (fichier).

Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHODS restantes ne doivent pas être exécutées. De plus, l'application EDD ne doit pas dessiner la WAVEFORM si une METHOD se termine pour une raison imprévue.

Structure lexicale

INIT_ACTIONS [(reference)<exp>]+, DEFINITION

Les attributs d'INIT_ACTIONS sont spécifiés au Tableau 167.

Tableau 167 – Attribut EXIT_ACTIONS, INIT_ACTIONS, REFRESH_ACTIONS

Usage	Attribut	Description
o	reference	Référence à une instance de METHOD
o	DEFINITION	Élément lexical (voir XXX)

7.29.2.6 KEY_POINTS**7.29.2.6.1 Structure générale****Objet**

L'attribut KEY_POINTS (points clés) spécifie les points clés dans la WAVEFORM qu'il convient de mettre en évidence à l'aide de l'application EDD. La façon selon laquelle ces points sont mis en surbrillance est définie par la spécification de l'EDD. Par défaut, il convient qu'une WAVEFORM sans attribut KEY_POINTS ne comporte aucun point mis en évidence.

Structure lexicale

KEY_POINTS [X_VALUES, Y_VALUES]

Les attributs de KEY_POINTS sont spécifiés au Tableau 168.

Tableau 168 – Attribut de KEY_POINTS

Usage	Attribut	Description
m	X_VALUES	Spécifie la coordonnée X de chaque KEY_POINT
m	Y_VALUES	Spécifie la coordonnée Y de chaque KEY_POINT

7.29.2.6.2 Attributs spécifiques

7.29.2.6.2.1 X_VALUES

Objet

L'attribut X_VALUES spécifie la coordonnée X de chaque point qui doit 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 [reference, double, float, integer, unsigned_integer]+

Les attributs de X_VALUES sont spécifiés au Tableau 169.

Tableau 169 – Attribut de X_VALUES, Y_VALUES

Usage	Attribut	Description
o	reference	Référence à une instance de VARIABLE, un membre d'une instance de RECORD, un élément d'une instance de VALUE_ARRAY, un élément d'une instance de REFERENCE_ARRAY, une instance de VALUE_ARRAY ou une instance de REFERENCE_ARRAY
o	double	Constante à virgule flottante à double précision
o	float	Constante à virgule flottante à simple précision
o	integer	Constante entière
o	unsigned_integer	Constante entière non signée

7.29.2.6.2.2 Y_VALUES

Objet

L'attribut Y_VALUES spécifie la coordonnée Y de chaque point à mettre en évidence. Chaque coordonnée Y spécifiée dans Y_VALUES doit correspondre à une coordonnée X spécifiée dans X_VALUES.

Structure lexicale

Y_VALUES [reference, double, float, integer, unsigned_integer]+

Les attributs de Y_VALUES sont spécifiés au Tableau 169.

7.29.2.7 LINE_COLOR

Objet

L'attribut LINE_COLOR spécifie la couleur qu'il convient d'utiliser pour afficher la WAVEFORM. Par défaut, une WAVEFORM sans attribut LINE_COLOR s'affiche en utilisant une couleur par défaut définie par l'application EDD.

Structure lexicale

LINE_COLOR (integer, reference)<exp>

Les couleurs sont spécifiées par l'intermédiaire d'une valeur RVB, comme défini dans Cascading Style Sheets, niveau 2.

Les attributs de LINE_COLOR sont spécifiés au Tableau 170.

Tableau 170 – Attribut de LINE_COLOR

Usage	Attribut	Description
s	integer	Couleur à utiliser pour afficher la WAVEFORM
s	reference	Référence à une instance de VARIABLE contenant la couleur à utiliser pour afficher la WAVEFORM.

7.29.2.8 LINE_TYPE**Objet**

L'attribut LINE_TYPE spécifie le type de données contenues dans la WAVEFORM. Il convient d'afficher chaque WAVEFORM avec le même attribut LINE_TYPE de la même façon (c'est-à-dire, couleur de trait, motif de trait, épaisseur de trait, etc.). Par défaut, une WAVEFORM sans attribut LINE_TYPE utilise le style de trait DATA.

Structure lexicale

LINE_TYPE ([DATA, n]*, LOW_LOW_LIMIT, LOW_LIMIT, HIGH_LIMIT, HIGH_HIGH_LIMIT) <exp>

Les attributs de LINE_TYPE sont spécifiés au Tableau 171.

Tableau 171 – Attributs de LINE_TYPE

Usage	Attribut	Description
s	DATA	Spécifie que la WAVEFORM contient des données
c	n	Entier croissant commençant à un, qui identifie des types de trait DATA distincts si plusieurs sont présents
s	LOW_LOW_LIMIT	Spécifie que la WAVEFORM contient une limite très basse
s	LOW_LIMIT	Spécifie que la WAVEFORM contient une limite basse
s	HIGH_LIMIT	Spécifie que la WAVEFORM contient une limite élevée
s	HIGH_HIGH_LIMIT	Spécifie que la WAVEFORM contient une limite très élevée
s	TRANSPARENT	Spécifie que seuls les points de données de la WAVEFORM sont visibles

7.29.2.9 REFRESH_ACTIONS**Objet**

Les REFRESH_ACTIONS sont exécutées chaque fois que des modifications sont apportées au X_AXIS ou Y_AXIS du GRAPH dans lequel la WAVEFORM est affichée.

Les METHODS spécifiées doivent être exécutées dans l'ordre dans lequel elles apparaissent. Si une METHOD se termine pour une raison imprévue, les METHODS restantes ne doivent pas être exécutées. De plus, les X_VALUES et Y_VALUES de la WAVEFORM doivent être restaurées à leurs valeurs originales par l'application EDD, et la WAVEFORM sur le GRAPH ne doit pas être redessinée.

Structure lexicale

REFRESH_ACTIONS [(reference) <exp>]+, DEFINITION

Les attributs de REFRESH_ACTIONS sont spécifiés au Tableau 167.

7.29.2.10 Y_AXIS

Objet

L'attribut Y_AXIS spécifie le mode d'affichage de l'axe Y de la WAVEFORM. Chaque WAVEFORM sur un GRAPH partage un X_AXIS commun mais chaque WAVEFORM sur un GRAPH a son propre Y_AXIS. Par défaut, l'affichage de l'axe Y d'un GRAPH sans attribut Y_AXIS est dérivé des données dans la WAVEFORM.

Structure lexicale

Y_AXIS (reference) <exp>

L'attribut de Y_AXIS est spécifié au Tableau 172.

Tableau 172 – Attribut d'Y_AXIS

Usage	Attribut	Description
m	reference	Référence à une instance d'AXIS

7.30 Attributs communs

7.30.1 DEFINITION

Objet

Une DEFINITION spécifie les actions à effectuer, et repose sur un langage de programmation procédural (ANSI C, par exemple). Dans les DEFINITIONS, des METHODS et des Builtins (voir Annexe B) peuvent être appelés. Des éléments de base de l'EDD peuvent être utilisés dans une DEFINITION.

Les éléments suivants doivent être pris en charge.

- Types de données de base (voir D.5.1).
- Matrices (voir D.5.1)
- Opérateurs arithmétiques (voir D.5.1)
- Instructions (voir D.5.1)
- Invocation de METHODS et de Builtins
- METHODS avec arguments (appel par valeur, appel par référence)

Un appel par valeur est utilisé pour transmettre une copie des données à la METHOD. Des changements de la copie dans la METHOD ne modifient pas les données originales.

L'appel par référence est utilisé pour accéder directement aux données. Les changements affectent les données originales.

Structure lexicale

DEFINITION compound-statement

L'attribut de DEFINITION est spécifié au Tableau 173.

Tableau 173 – Attribut de DEFINITION

Usage	Attribut	Description
o	compound-statement	Spécifie un bloc d'instructions pour un langage de programmation procédural pris en charge

7.30.2 HELP

Objet

HELP spécifie du texte, qui donne une description de la construction.

NOTE Ce texte est destiné à être utilisé par des applications EDD pour information des utilisateurs.

Structure lexicale

HELP (string)<exp>

L'attribut de HELP est spécifié au Tableau 174.

Tableau 174 – Attribut HELP

Usage	Attribut	Description
m	string	Spécifie le texte lisible par l'utilisateur

7.30.3 LABEL

Objet

LABEL spécifie la désignation affichée d'un élément.

Structure lexicale

LABEL (string)<exp>

L'attribut de LABEL est spécifié au Tableau 175.

Tableau 175 – Attribut LABEL

Usage	Attribut	Description
m	string	Spécifie le texte lisible par l'utilisateur

7.30.4 MEMBERS

Objet

L'attribut MEMBERS définit les membres d'une construction de base. Chaque membre spécifie un élément du groupe et est défini par un ensemble de quatre attributs (identifier, reference, description, help).

Structure lexicale

MEMBERS [(identifier, reference, description, help)<exp>]+

Les attributs de MEMBERS sont spécifiés au Tableau 176.

Tableau 176 – Attributs de MEMBERS

Usage	Attribut	Description
m	identifier	Spécifie le nom par lequel l'élément peut être référencé
m	reference	Référence à un élément EDD. Le type de la référence dépend des constructions de base Si MEMBERS est un attribut de CHART, l'attribut reference fait référence à une instance de SOURCE Si MEMBERS est un attribut de COLLECTION, l'attribut reference fait référence à une instance d'AXIS, BLOCK_A, BLOCK_B, CHART, COLLECTION, COMMAND, CONNECTION, DOMAIN, EDIT_DISPLAY, FILE, GRAPH, LIST, MENU, METHOD, PROGRAM, RECORD, REFERENCE_ARRAY, REFRESH, RESPONSE_CODES, SOURCE, UNIT, VALUE_ARRAY, VARIABLE, VARIABLE_LIST, WAVEFORM ou WRITE_AS_ONE. Si MEMBERS est un attribut de FILE ou LIST, la référence est une référence à une instance de COLLECTION, LIST, RECORD, REFERENCE_ARRAY, VALUE_ARRAY ou VARIABLE. Pour une COLLECTION, le type de la COLLECTION doit être LIST, RECORD, REFERENCE_ARRAY, VALUE_ARRAY ou VARIABLE Si MEMBERS est un attribut de GRAPH, l'attribut reference fait référence à une instance de WAVEFORM Si MEMBERS est un attribut de RECORDS, reference fait référence à une instance de VARIABLE Si MEMBERS est un attribut de SOURCE, reference fait référence à une instance de VARIABLE ou un membre d'une instance de RECORD Si MEMBERS est un attribut de VARIABLE_LISTS, reference fait référence à une instance de RECORDS, VALUE_ARRAY ou VARIABLE
o	description	Brève description de l'élément
o	help	Spécifie le texte d'aide pour l'élément.

7.30.5 RESPONSE_CODES

Objet

Les RESPONSE_CODES (codes de réponse) spécifient les valeurs qu'un dispositif peut retourner en tant qu'information d'erreur. Chaque VARIABLE, RECORD, VALUE_ARRAY, VARIABLE_LIST, PROGRAM, COMMAND ou DOMAIN peut avoir son propre ensemble de RESPONSE_CODES.

Deux structures lexicales pour l'attribut RESPONSE_CODES sont fournies:

- le RESPONSE_CODES référencé contient une référence à une instance de RESPONSE_CODES;
- le RESPONSE_CODES incorporé permet de spécifier les éléments RESPONSE_CODES directement à l'instance d'élément.

Structure lexicale pour les RESPONSE_CODES référencés

RESPONSE_CODES (reference) <exp>

L'attribut du RESPONSE_CODES référencé est spécifié au Tableau 177.

Tableau 177 – Attribut de RESPONSE_CODES

Usage	Attribut	Description
m	reference	Fait référence à une instance de RESPONSE_CODES

Structure lexicale pour les RESPONSE_CODES incorporés

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

Les attributs de RESPONSE_CODES incorporés sont spécifiés au Tableau 129.

7.31 Redirection de la sortie (OPEN et CLOSE)

Objet

La sortie du processeur EDDL peut être écrite dans des fichiers de sortie. Le mot clé OPEN indique au processeur EDDL de créer un fichier de sortie distinct pour les constructions EDD et le mot clé CLOSE de fermer de fichier de sortie. Une tentative d'ouverture d'un fichier de sortie déjà ouvert donne lieu à une erreur. Plusieurs ouvertures et fermetures du même fichier de sortie dans une EDD amènent le processeur à écraser la totalité du fichier à chaque ouverture.

- Le mot clé OPEN ouvre un fichier de sortie en mode d'écrasement. À chaque traitement d'un mot clé OPEN, un fichier de sortie est ouvert, et crée le cas échéant, sauf si le fichier est déjà ouvert, auquel cas une erreur se produit. Tous les objets générés sont écrits dans tous les fichiers ouverts.
- Le mot clé CLOSE empêche l'écriture d'une autre sortie dans le fichier ouvert. Si un fichier est ouvert, tous les objets suivants écrasent le fichier.

Structure lexicale

```
OPEN filename, [(construct) <exp>]+
CLOSE filename
```

Les attributs sont spécifiés au Tableau 178.

Tableau 178 – Attributs d'OPEN et de CLOSE

Usage	Attribut	Description
M	filename	Chaîne de lettres ou de chiffres indiquant le nom du système de fichiers
M	construct	Liste de constructions écrites dans un fichier de sortie distinct

7.32 Expression conditionnelle

Objet

Des expressions conditionnelles sont utilisées pour déclarer des valeurs alternatives ou calculer des valeurs pour un attribut d'un élément de base. La valeur de l'expression est évaluée pendant le temps d'exécution. L'instruction conditionnelle IF ou SELECT peut être utilisée pour activer ou désactiver des instances référencées d'une liste d'attributs ITEMS de MENU, etc.

Trois types d'expressions conditionnelles peuvent être utilisés.

- Une instruction conditionnelle IF est utilisée pour spécifier un attribut qui a deux définitions alternatives. L'expression spécifiée dans l'instruction conditionnelle IF est évaluée. Si le résultat est différent de zéro, l'attribut est spécifié par une clause then-clause ("alors"); sinon, il est spécifié par une clause else-clause ("sinon").
- Une instruction conditionnelle SELECT est utilisée pour spécifier un attribut qui a plusieurs définitions alternatives. L'expression select-expression est évaluée et comparée à chaque integer_n de CASE (cas). Si une correspondance est trouvée, la select-clause appropriée est valide. Si aucun integer_n correspondant n'est trouvé, la clause par défaut default-clause suivant DEFAULT est utilisée.

- Une expression primaire, unaire ou binaire (voir 7.34.7). Ce type d'expression conditionnelle peut être utilisé uniquement pour des valeurs numériques (par exemple, DEFAULT_VALUE, MIN_VALUE, SLOT, INDEX).

EXEMPLE Plusieurs valeurs peuvent être définies pour une DEFAULT_VALUE, MIN_VALUE, etc. en utilisant une instruction conditionnelle IF ou SELECT.

Structure lexicale pour l'instruction conditionnelle IF

IF (if-expression), (then-clause)+ ELSE (else-clause)+

Les attributs sont spécifiés au Tableau 179.

Structure lexicale pour l'instruction conditionnelle SELECT

SELECT select-expression, (CASE integer_n, select-clause)+, (DEFAULT, default-clause)

Les attributs sont spécifiés au Tableau 179.

Tableau 179 – Instruction conditionnelle IF, SELECT

Usage	Attribut	Description
m	if-expression	Évaluée pour déterminer si la clause then-clause ou else-clause est utilisée pour définir un attribut. Si l'expression if-expression contient une référence à une instance de VARIABLE, la valeur de la VARIABLE référencée est utilisée
m	then-clause	Définition pour l'attribut si la valeur de condition est différente de zéro. La structure de clause dépend de l'attribut défini. Elle peut également prendre la forme d'une autre instruction conditionnelle
o	else-clause	Définition de l'attribut si la valeur de condition est différente de zéro. La structure de clause dépend de l'attribut défini. Elle peut également prendre la forme d'une autre instruction conditionnelle
m	select-expression	Évalué pour déterminer la clause select-clause utilisée pour définir un attribut. Si l'expression select-expression contient une référence à une instance de VARIABLE, la valeur de la VARIABLE référencée est utilisée
m	integer_n	Spécifie un numéro identifiant le CASE correspondant. Si un entier correspond à integer_n, la select_clause correspondante est utilisée
m	select-clause	Définition de l'attribut pour chaque CASE, et définition DEFAULT. La structure de chaque clause dépend de l'attribut défini. Quel que soit l'attribut, chaque clause peut également prendre la forme d'une autre instruction conditionnelle
o	default-clause	Définition de la définition DEFAULT. La structure de default-clause dépend de l'attribut défini. Quel que soit l'attribut, chaque clause peut également prendre la forme d'une autre instruction conditionnelle

Si la clause

- then-clause,
- else-clause,
- select-clause,
- default-clause

contient des références à des instances de METHOD ou des Builtins, la valeur retournée de la METHOD ou du Builtin référencé doit être utilisée dans la clause. Les valeurs de retour doivent être compatibles au niveau du type.

7.33 Référencement

7.33.1 Référencement d'une instance d'EDD

Objet

Des références sont utilisées dans une EDD pour faire référence à d'autres éléments de l'EDD.

EXEMPLE Un élément PRE_EDIT_ACTIONS de VARIABLE fait référence à des instances de METHOD définies ailleurs dans l'EDD.

Structure lexicale

L'attribut est spécifié au Tableau 180.

Tableau 180 – Référencement d'une instance d'EDD

Usage	Attribut	Description
m	identifier	Identifiant d'une instance d'EDD

7.33.2 Référencement de bits d'une VARIABLE BIT ENUMERATED

Objet

Ce référencement est utilisé pour accéder à un bit d'une instance VARIABLE BIT ENUMERATED.

Structure lexicale

variable-reference, bit-mask

Les attributs sont spécifiés au Tableau 181.

Tableau 181 – Référencement d'éléments de VARIABLE

Usage	Attribut	Description
m	variable-reference	Référence à une instance de VARIABLE; le type de l'instance de VARIABLE doit être BIT ENUMERATED
m	member-identifier	Masque binaire qui identifie un seul bit dans la variable. Il s'agit du masque utilisé pour définir le bit dans la VARIABLE

7.33.3 Référencement des membres d'un RECORD

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de RECORD.

Structure lexicale

record-reference, member-identifier

Les attributs sont spécifiés au Tableau 182.

Tableau 182 – Référencement d'éléments de RECORD

Usage	Attribut	Description
m	record-reference	Référence à une instance de RECORD
m	member-identifier	Identifiant d'un élément de MEMBERS

7.33.4 Référencement d'éléments d'un VALUE_ARRAY

Objet

Ce référencement est utilisé pour accéder à un élément d'une instance de VALUE_ARRAY.

Structure lexicale

value-array-reference, expression

Les attributs sont spécifiés dans le Tableau 183.

Tableau 183 – Référencement d'éléments de VALUE_ARRAY

Usage	Attribut	Description
m	value-array-reference	Référence à une instance de VALUE_ARRAY
m	expression	Expression qui doit être évaluée à un entier non négatif compris dans l'intervalle d'index de l'attribut VALUE_ARRAY (pour une description des expressions, voir 7.35).

7.33.5 Référencement des membres d'une COLLECTION

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de COLLECTION.

Structure lexicale

collection-reference, member-identifiant

Les attributs sont spécifiés au Tableau 184.

Tableau 184 – Référencement des membres de COLLECTION

Usage	Attribut	Description
m	collection-reference	Référence à une instance de COLLECTION
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.6 Référencement d'éléments d'un REFERENCE_ARRAY

Objet

Ce référencement est utilisé pour accéder à un élément d'une instance de REFERENCE_ARRAY.

Structure lexicale

reference-array-reference, expression

Les attributs sont spécifiés au Tableau 185.

Tableau 185 – Référencement des membres de REFERENCE_ARRAY

Usage	Attribut	Description
m	reference-array-reference	Référence à une instance de REFERENCE_ARRAY
m	expression	Expression qui doit être évaluée à un entier non négatif compris dans l'intervalle d'index du REFERENCE_ARRAY (pour la description des expressions, voir 7.34.7).

7.33.7 Référencement des membres d'une VARIABLE_LISTS

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de VARIABLE_LISTS.

Structure lexicale

variable-list-reference, member-identifiant

Les attributs sont spécifiés au Tableau 186.

Tableau 186 – Référencement des membres de VARIABLE_LISTS

Usage	Attribut	Description
m	variable-list-reference	Référence à une instance de VARIABLE_LISTS
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.8 Référencement d'éléments BLOCK_A PARAMETERS

Objet

Ce référencement est utilisé pour accéder à un élément de l'attribut PARAMETERS d'une instance de BLOCK_A.

Structure lexicale

PARAM, member-identifiant

Les attributs sont spécifiés au Tableau 187.

Tableau 187 – Référencement des membres de BLOCK_A PARAMETERS

Usage	Attribut	Description
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.9 Référencement d'éléments de BLOCK_A PARAMETER_LISTS

Objet

Ce référencement est utilisé pour accéder à un élément de l'attribut PARAMETER_LISTS d'une instance de BLOCK_A.

Structure lexicale

PARAM_LIST, member-identifiant

Les attributs sont spécifiés au Tableau 188.

Tableau 188 – Référencement des membres de BLOCK_A PARAMETER_LISTS

Usage	Attribut	Description
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.10 Référencement des membres de BLOCK_A LOCAL_PARAMETERS

Objet

Ce référencement est utilisé pour accéder à un élément de l'attribut LOCAL_PARAMETERS d'une instance de BLOCK_A.

Structure lexicale

LOCAL_PARAM, member-identifiant

Les attributs sont spécifiés au Tableau 189.

Tableau 189 – Référencement des membres de BLOCK_A LOCAL_PARAMETER

Usage	Attribut	Description
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.11 Référencement de BLOCK_A CHARACTERISTICS

Objet

Ce référencement est utilisé pour accéder aux CHARACTERISTICS (caractéristiques) d'une instance de BLOCK_A.

Structure lexicale

BLOCK, characteristics-identifiant

L'attribut est spécifié au Tableau 190.

Tableau 190 – Référencement de BLOCK_A CHARACTERISTICS

Usage	Attribut	Description
m	characteristics-identifiant	Identifiant de la référence de CHARACTERISTICS

7.33.12 Référencement des membres d'un FILE

Objet

Ce référencement est utilisé pour accéder à un membre d'une instance de FILE (fichier).

Structure lexicale

file-reference, member-identifiant

Les attributs sont spécifiés au Tableau 191.

Tableau 191 – Référencement des membres de FILE

Usage	Attribut	Description
m	file-reference	Référence à une instance de FILE
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.13 Référencement d'éléments de LIST

Objet

Ce référencement est utilisé pour accéder à un élément d'une instance de LIST.

Structure lexicale

list-reference, (expression, FIRST, LAST)

Les attributs sont spécifiés au Tableau 192.

Tableau 192 – Référencement d'éléments de LIST

Usage	Attribut	Description
m	list-reference	Référence à une instance de LIST
s	expression	Expression qui doit être évaluée à un entier non négatif compris dans l'intervalle d'index de LIST (pour la description des expressions, voir 7.34.7).
s	FIRST	Référence au premier élément de LIST
s	LAST	Référence au dernier élément de LIST

7.33.14 Référencement des membres de CHART**Objet**

Ce référencement est utilisé pour accéder à un membre d'une instance de CHART (diagramme).

Structure lexicale

chart-reference, member-identifiant

Les attributs sont spécifiés au Tableau 193.

Tableau 193 – Référencement des membres de CHART

Usage	Attribut	Description
m	chart-reference	Référence à une instance de CHART
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.15 Référencement de membres d'un GRAPH**Objet**

Ce référencement est utilisé pour accéder à un membre d'une instance de GRAPH.

Structure lexicale

graph-reference, member-identifiant

Les attributs sont spécifiés au Tableau 194.

Tableau 194 – Référencement des membres de GRAPH

Usage	Attribut	Description
m	graph-reference	Référence à une instance de GRAPH
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.16 Référencement des membres d'une SOURCE**Objet**

Ce référencement est utilisé pour accéder à un membre d'une instance de GRAPH.

Structure lexicale

source-reference, member-identifiant

Les attributs sont spécifiés au Tableau 195.

Tableau 195 – Référencement des membres de SOURCE

Usage	Attribut	Description
m	source-reference	Référence à une instance de SOURCE
m	member-identifiant	Identifiant d'un élément de MEMBERS

7.33.17 Référencement d'AXIS de GRAPH, SOURCE, WAVEFORM**Objet**

Ce référencement est utilisé pour accéder à un AXIS (axe) de GRAPH (graphique), SOURCE ou WAVEFORM (courbe).

Structure lexicale

reference, (X_AXIS, Y_AXIS)

Les attributs sont spécifiés au Tableau 196.

Tableau 196 – Référencement d'AXIS d'un GRAPH, SOURCE, WAVEFORM

Usage	Attribut	Description
m	reference	Référence à une instance de GRAPH, de SOURCE ou de WAVEFORM
c/o	X_AXIS	c utilisé uniquement avec GRAPH; Y_AXIS utilisé dans tous les autres cas
c/o	Y_AXIS	c utilisé uniquement avec SOURCE et WAVEFORM; X_AXIS utilisé dans tous les autres cas

7.34 Chaînes**7.34.1 Spécification d'une chaîne sous forme de littéral chaîne****Objet**

Une chaîne de texte peut être spécifiée sous la forme d'un littéral chaîne (voir A.2.3).

Structure lexicale

string

L'attribut est spécifié au Tableau 197.

Tableau 197 – Chaîne sous la forme d'un littéral chaîne

Usage	Attribut	Description
m	string	Spécifie un littéral chaîne

7.34.2 Spécification d'une chaîne en tant que variable chaîne**Objet**

Une chaîne de texte spécifiée en tant que variable chaîne est la valeur de la variable chaîne.

Structure lexicale

reference

L'attribut est spécifié au Tableau 198.

Tableau 198 – Chaîne sous la forme d'une variable chaîne

Usage	Attribut	Description
m	reference	Référence à une instance de VARIABLE

7.34.3 Spécification d'une chaîne en tant que valeur d'énumération

Objet

Une chaîne de valeur d'énumération est une chaîne associée à une des valeurs d'une variable d'énumération.

Structure lexicale

reference, value

Les attributs sont spécifiés au Tableau 199.

Tableau 199 – Chaîne en tant que valeur d'énumération

Usage	Attribut	Description
m	reference	Référence à une instance de VARIABLE de TYPE ENUMERATION
m	value	Valeur correspondante du texte de description dans la liste d'énumération

Exemple La chaîne de valeur d'énumération suivante spécifie la chaîne associée à la valeur 4 de la variable units_code: units_code (4)

7.34.4 Spécification d'une chaîne en tant que référence de dictionnaire

Objet

Une référence de dictionnaire pointe vers une chaîne dans un dictionnaire de texte.

Structure lexicale

reference

L'attribut est spécifié au Tableau 200.

Tableau 200 – Chaîne en tant que référence de dictionnaire

Usage	Attribut	Description
m	reference	Référence à une ou plusieurs chaînes spécifiées dans un dictionnaire (voir 7.36)

7.34.5 Référencement d'attributs HELP et LABEL d'instances d'EDD

Objet

Pour obtenir la chaîne actuelle de LABEL ou HELP d'une instance d'EDD, l'instance et l'attribut LABEL ou HELP doivent être référencés.

Structure lexicale

reference, [HELP, LABEL]

Les attributs sont spécifiés au Tableau 201.

Tableau 201 – Référencement d'attributs HELP et LABEL d'instances d'EDD

Usage	Attribut	Description
m	reference	Référence à une instance d'EDD contenant les attributs LABEL et/ou HELP
s	HELP	Fournit la chaîne HELP de l'instance
s	LABEL	Fournit la chaîne LABEL de l'instance

7.34.6 Opérations de chaîne

Objet

Les chaînes littérales, les variables chaînes, les valeurs d'énumération chaîne, les références de dictionnaire et les valeurs d'attribut de constructions de base peuvent être concaténées les unes avec les autres. Si un terme contient des références de types de données numériques, les valeurs doivent être converties en chaîne grâce à l'application EDD.

EXEMPLE `variable_of_type_string = "Description"`
`"Electronic" + "Device" + variable_of_type_string` donne `"Electronic Device Description"`.

Structure lexicale

`(string)<exp>, [(concatenation-operator, string)<exp>]+`

Les attributs sont spécifiés au Tableau 202

Tableau 202 – Opération de chaîne

Usage	Attribut	Description
m	string	Chaînes littérales, variables chaînes, valeurs d'énumération de chaîne, références de dictionnaire ou référence de HELP, LABEL
o	concatenation-operator	Spécifie que les chaînes doivent être concaténées

7.34.7 Formats de chaîne d'invite de commande

Objet

Des chaînes d'invite de commande peuvent être utilisées pour incorporer des éléments formatés dans des chaînes. Ces éléments sont composés d'un spécificateur de format facultatif et d'un VARIABLE ID ou une référence.

Structure lexicale

`format, (reference, id)`
`label, (reference, id)`

Les attributs sont spécifiés au Tableau 203.

Tableau 203 – Spécificateur de format

Usage	Attribut	Description
m	format	Composé de la chaîne de format printf ANSI C (voir l'ISO/CEI 9899). Si le format n'est pas spécifié, le format de l'instance de VARIABLE référencée est utilisé. Si aucun format n'est spécifié, un format par défaut est utilisé en fonction du type d'instance de VARIABLE
m	label	Spécifie le LABEL de l'instance de VARIABLE
o/s	reference	Spécifie la valeur ou l'étiquette de l'instance de VARIABLE référencée
o/s	id	Spécifie les ID d'élément des VARIABLES

7.35 Expression

7.35.1 Structure générale

Objet

Il existe trois types d'expressions.

- Les expressions primaires comprennent des constantes, des expressions entre parenthèses, des références de VARIABLE, des accès d'élément d'index ou un accès au statut d'instances de VARIABLE. Ces opérations sont associatives par la gauche, c'est-à-dire qu'elles sont évaluées de gauche à droite.
- Des expressions unaires comprennent tous les opérateurs évaluant un opérande unique. Ces opérations sont associatives par la droite, c'est-à-dire qu'elles sont évaluées de droite à gauche.
- Les expressions binaires comprennent tous les opérateurs évaluant deux opérandes. Ces opérations sont associatives par la gauche, c'est-à-dire qu'elles sont évaluées de gauche à droite.

7.35.2 Expressions primaires

Le Tableau 204 ci-dessous résume les expressions primaires dans le langage EDDL.

Tableau 204 – Expressions primaires

Expression primaire	Description
constante	Expression dont la valeur est identique à la valeur de la constante
expression entre parenthèses	Expression dont la valeur est identique à la valeur de l'expression incluse
référence de variable	Expression dont la valeur est celle de la VARIABLE référencée
ARRAY_INDEX	Expression dont la valeur est dérivée du numéro d'index du VALUE_ARRAY référençant
valeurs d'attribut de VARIABLEs	Référence composée de l'identifiant d'une instance de VARIABLE et d'un attribut. La référence fournit la valeur correspondante de l'attribut (voir Tableau 205 pour les valeurs retournées)
CURRENT_ROLE	Spécifie une constante chaîne qui contient le rôle actuel. La valeur de CURRENT_ROLE (rôle actuel) doit être prise en charge par l'application EDD. Les valeurs de la constante chaîne doivent être définies par l'attribut ROLE de MENU
PI	Spécifie la constante mathématique π
valeurs d'attribut d'un AXIS	Référence à une instance et un attribut d'AXIS. La référence fournit la valeur correspondante des attributs (voir Tableau 206 pour les valeurs retournées). Ces attributs peuvent uniquement être lus, sauf indication contraire
valeurs d'attribut d'une LIST	Référence à une instance et un attribut de LIST. La référence fournit la valeur correspondante des attributs (voir Tableau 207 pour les valeurs retournées). Ces attributs peuvent uniquement être lus, sauf indication contraire

Tableau 205 – Valeurs d'attribut de VARIABLE

Attributs	Type de données	Description
MIN_VALUE	type d'instance de VARIABLE	Renvoie une valeur qui satisfait à la MIN_VALUE de l'instance de VARIABLE. Si plus d'une MIN_VALUE est définie, MIN_VALUE plus l'identifiant d'intervalle (voir Tableau 141) doit être utilisé pour le référencement
MAX_VALUE	type d'instance de VARIABLE	Renvoie une valeur qui satisfait à la MAX_VALUE de l'instance de VARIABLE. Si plusieurs MAX_VALUE sont définies, MAX_VALUE plus l'identifiant d'intervalle (voir Tableau 141) doit être utilisé pour le référencement
SCALING_FACTOR	double	Renvoie une valeur qui satisfait à la SCALING_FACTOR de l'instance de VARIABLE
DEFAULT_VALUE	type d'instance de VARIABLE	Renvoie une valeur qui satisfait à la DEFAULT_VALUE de l'instance de VARIABLE
INITIAL_VALUE	type d'instance de VARIABLE	Renvoie une valeur qui satisfait à l'INITIAL_VALUE de l'instance de VARIABLE
VARIABLE_STATUS		Renvoie une valeur qui indique le statut de la VARIABLE. Les statuts suivants sont définis : VARIABLE_STATUS_NONE spécifie que la valeur est valide et pas l'un des autres statuts VARIABLE_STATUS_INVALID spécifie que la valeur est hors de l'intervalle VARIABLE_STATUS_NOT_ACCEPTED spécifie que la valeur n'a pas pu être transférée VARIABLE_STATUS_NOT_SUPPORTED – spécifie que la valeur n'est pas prise en charge VARIABLE_STATUS_CHANGED spécifie que la valeur a été modifiée par l'utilisateur ou par une méthode VARIABLE_STATUS_LOADED spécifie que la valeur a été chargée avec succès VARIABLE_STATUS_INITIAL spécifie que la valeur est inchangée VARIABLE_STATUS_NOT_CONVERTED spécifie que la valeur n'a pas été convertie, par exemple, parce que les unités ne sont pas compatibles

Tableau 206 – Valeurs d'attribut d'AXIS

Attributs	Type de données	Description
MIN_VALUE	type arithmétique	Renvoie une valeur qui satisfait à la MIN_VALUE de l'instance d'AXIS
MAX_VALUE	type arithmétique	Renvoie une valeur qui satisfait à la MAX_VALUE de l'instance d'AXIS
SCALING	LINEAR (linéaire) ou LOGARITHMIC (logarithmique)	Renvoie une valeur qui satisfait à la SCALING de l'instance d'AXIS
VIEW_MIN	type arithmétique	Renvoie une valeur qui satisfait au VIEW_MIN de l'instance d'AXIS. VIEW_MIN peut être lu et écrit
VIEW_MAX	type arithmétique	Renvoie une valeur conforme au VIEW_MAX de l'instance d'AXIS. VIEW_MAX peut être lu et écrit

Tableau 207 – Valeurs d'attribut de LIST

Attributs	Type de données	Description
CAPACITY	unsigned_integer	Renvoie une valeur qui satisfait à la CAPACITY de l'instance de LIST
COUNT	unsigned_integer	Renvoie une valeur qui satisfait à la COUNT de l'instance de LIST

7.35.3 Expressions unaires

Une expression unaire est constituée d'un opérande numérique, une expression, précédée par un opérateur unaire. Le Tableau 208 spécifie les expressions unaires dans le langage EDDL.

Tableau 208 – Expressions unaires

Opérateur	Description
++	Incrémente son opérande en ajoutant la valeur 1 à la valeur de l'opérande. L'opérateur d'incrément peut être utilisé en tant qu'opérateur préfixe ou postfixe. Les deux notations préfixe et postfixe génèrent un incrément de l'opérande. La différence est que l'expression préfixe modifie l'opérande avant d'utiliser sa valeur, alors que l'expression postfixe modifie l'opérande après avoir utilisé sa valeur. Par conséquent, les significations du résultat de ces expressions peuvent être différentes selon le contexte du programme
--	Décrémente son opérande en retirant la valeur 1 de l'opérande. L'opérateur de décrémentation peut être utilisé en tant qu'opérateur préfixe ou postfixe. Les deux notations préfixe et postfixe génèrent un décrémentation de l'opérande. La différence est que l'expression préfixe modifie l'opérande avant d'utiliser sa valeur, alors que l'expression postfixe modifie l'opérande après avoir utilisé sa valeur. Par conséquent, les significations du résultat de ces expressions peuvent être différentes selon le contexte du programme
-	Négation arithmétique de l'opérande
~	Négation binaire de l'opérande, c'est-à-dire que chaque bit du résultat est l'inverse du bit correspondant de l'opérande. L'opérande de l'opérateur ~ doit avoir une valeur entière
!	Négation logique de l'opérande

7.35.4 Expressions binaires

7.35.4.1 Structure générale

Une expression binaire est constituée de deux opérandes ou expressions, séparés par un opérateur binaire. Si un opérande a une valeur à virgule flottante, l'autre opérande est converti (promu) en valeur à virgule flottante. Cette sous-section spécifie les types d'expressions binaires suivants.

- Multiplicatif
- Additif
- Décalage
- Relationnel
- Égalité
- ET binaire (&)
- XOU binaire (^)
- OU binaire (|)
- ET logique (&&)
- OU logique (||)
- Évaluation conditionnelle
- Affectations

7.35.4.2 Opérateurs multiplicatifs

Les opérateurs multiplicatifs spécifient la multiplication et la division d'opérandes. Le Tableau 209 spécifie les opérateurs multiplicatifs.

Tableau 209 – Opérateurs multiplicatifs

Opérateur	Description
*	Spécifie la multiplication de ses opérandes
/	Spécifie la division du premier opérande par le deuxième opérande. Le résultat de l'opérateur / est le quotient de la division
%	Spécifie la division du premier opérande par le deuxième opérande. Le résultat de l'opérateur % est le résidu

7.35.4.3 Opérateurs additifs

Les opérateurs additifs spécifient l'addition et la soustraction d'opérandes numériques. Si cet opérateur est appliqué à des opérandes numériques, le résultat est alors la somme de ses deux opérandes, et l'opération est commutative. Si l'opérateur + est appliqué à des opérandes chaînes, le résultat est une valeur de chaîne générée par ajout d'une seconde chaîne à la première chaîne et l'opération n'est pas commutative.

L'opérateur moins indique une soustraction. La valeur résultante de cette opération est la différence entre ses opérandes. Le second opérande est soustrait du premier. Le Tableau 210 spécifie des opérateurs additifs.

Tableau 210 – Opérateurs additifs

Opérateur	Description
+	Spécifie l'addition de ses opérandes
-	Spécifie la soustraction du second opérande du premier

7.35.4.4 Opérateurs de décalage

Les opérateurs << et >> spécifient un décalage du premier opérande du nombre de bits spécifié par le second opérande. Les opérandes des opérateurs << et >> doivent avoir des valeurs entières. Le Tableau 211 spécifie les opérateurs de décalage.

Tableau 211 – Opérateurs de décalage

Opérateur	Description
<<	Décale le premier opérande vers la gauche. Les bits décalés à l'extérieur sont rejetés, et les bits laissés vacants sont remplis avec des zéros
>>	Décale le premier opérande vers la droite. Les bits décalés à l'extérieur sont rejetés. Si le premier opérande est inférieur à 0, les bits laissés vacants sont remplis avec des 1, sinon ils sont remplis avec des zéros

7.35.4.5 Opérateurs relationnels

Les opérateurs relationnels (<, <=, >, >=) spécifient une comparaison des opérandes. Le résultat de ce type d'expression est 1 si la relation testée est vraie; sinon, le résultat est 0. Les opérateurs doivent être du même type de données. Les opérandes doivent être des types de données numériques. Le Tableau 212 spécifie les opérateurs relationnels.

Tableau 212 – Opérateurs relationnels

Opérateur	Description
<	Soumet à essai la relation "inférieur à"
<=	Soumet à essai la relation "inférieur ou égal à"
>	Soumet à essai la relation "supérieur à"
>=	Soumet à essai la relation "supérieur ou égal à"

7.35.4.6 Opérateurs d'égalité

Les opérateurs d'égalité sont == et !=. Le résultat de ce type d'expression est 1 si la relation testée est vraie; sinon, le résultat est 0. Les opérateurs doivent être du même type de données. Les types de données de tous les opérateurs doivent être identiques. Tous les types de données sont admis. Le Tableau 213 spécifie les opérateurs d'égalité.

Tableau 213 – Opérateurs d'égalité

Opérateur	Description
==	Soumet à essai la relation "égal"
!=	Soumet à essai la relation "n'est pas égal"

7.35.4.7 Opérateur ET binaire (&)

L'opérateur & spécifie le ET binaire de ses opérandes entiers, c'est-à-dire que chaque bit du résultat est défini si chacun des bits correspondants des opérandes est défini. Les opérandes de l'opérateur & doivent avoir des valeurs entières.

7.35.4.8 Opérateur XOU binaire (^)

L'opérateur ^ spécifie le OU exclusif binaire de ses opérandes entiers, c'est-à-dire que chaque bit du résultat est défini si un seul des bits correspondants des opérandes est défini. Les opérandes de l'opérateur ^ doivent avoir des valeurs entières.

7.35.4.9 Opérateur OU binaire (|)

L'opérateur | spécifie le OU inclusif binaire de ses opérandes entiers, c'est-à-dire que chaque bit du résultat est défini si l'un des bits correspondants des opérandes est défini. Les opérandes de l'opérateur | doivent avoir des valeurs entières.

7.35.4.10 Opérateur ET logique (&&)

L'opérateur && spécifie l'évaluation ET booléenne de ses opérandes entiers. Le résultat de ce type d'expression est 1 si les opérandes ne sont pas égales à 0. Sinon, le résultat est 0. Si la première opérande est égale à 0, la deuxième n'est pas évaluée.

7.35.4.11 Opérateur OU logique (||)

L'opérateur || spécifie l'évaluation OU booléenne de ses opérandes entiers. Le résultat de ce type d'expression est 1 si les opérandes ne sont pas égales à 0. Sinon, le résultat est 0. Si la première opérande n'est pas égale à 0, la deuxième n'est pas évaluée.

7.35.4.12 Évaluation conditionnelle

Les opérateurs ? et: sont utilisés pour l'évaluation conditionnelle.

EXEMPLE 1

```
result = logexpr ? trueexpr: falseexpr;
```

logexpr est évalué et, s'il est différent de zéro, le résultat est la valeur de trueexpr. Sinon, falseexpr est exécuté.

EXEMPLE 2 La structure if-else suivante a une logique identique à l'exemple 1:

```
if (logexpr)
    result = trueexpr;
else
    result = falseexpr;
```

7.35.4.13 Affectations

Les langages de programmation procéduraux offrent une série d'opérateurs d'affectation, tous étant associatifs. Tous les opérateurs d'affectation requièrent une expression unaire comme opérande de gauche. L'opérande de droite peut être de nouveau une expression d'affectation. Le type d'expression d'affectation correspond à son opérande de gauche. La valeur d'une opération d'affectation est celle stockée dans l'opérande de gauche après l'affectation.

L'opérateur = binaire indique l'affectation simple. Les opérateurs binaires *=, /=, %=, +=, -=, >=>, <=<=, &=, ^= et |= indiquent une expression d'affectation composée.

Une affectation composée de forme générale, telle que

```
expr1 <operator>= expr2
```

équivalut à l'expression

```
expr1 = expr1 <operator> (expr2)
```

Le type de données d'expr2 doit être convertible en type de données d'expr1. La sémantique des affectations est spécifiée en 7.35.2, 7.35.3 et 7.35.4.

7.35.4.14 Liste d'expressions

Chaque expression peut être composée d'une liste d'expressions binaires séparées par une virgule. Une paire d'expressions séparées par une virgule est évaluée de gauche à droite, et la valeur de l'expression de gauche est ignorée. Le type et la valeur du résultat sont ceux de l'opérande de droite.

EXEMPLE 1 c -= (a=5, b=3-a, c=b+a, a*=b+=4+2*a);
où les valeurs 60, 12 et -57 sont stockées dans les variables a, b et c, respectivement.

EXEMPLE 2 fct (x, (y=8, y*25.4), z)
comportent trois arguments, dont le deuxième prend la valeur 203.2. Dans les contextes où un sens particulier est donné à la virgule (dans une liste de paramètres de fonction réels et d'initialiseurs, par exemple), l'opérateur de virgule peut uniquement apparaître entre parenthèses (la fonction d'appel, par exemple)

7.36 Dictionnaire de texte

Structure générale

Le dictionnaire est une collection de chaînes de textes qui peuvent être utilisées dans une EDD. Le texte du dictionnaire est référencé dans une EDD. On peut accéder au dictionnaire depuis n'importe quelle EDD afin d'assurer la cohérence des chaînes de texte (par exemple, pour utilisation dans des attributs de LABEL et HELP).

Généralement, un dictionnaire de texte commun est utilisé pour un groupe de dispositifs types. De plus, un dictionnaire spécifique pour l'utilisateur peut être spécifié. Le dictionnaire de texte prend en charge des textes multilingues.

Structure lexicale

```
identifier (language-code, country-code, string)+
```

Les attributs sont spécifiés au Tableau 214.

Tableau 214 – Attributs de dictionnaire de texte

Usage	Attribut	Description
m	Identifiant	Permet de référencer la chaîne de texte dans une EDD
m	language-code	Spécifie la langue utilisée et la présentation ASCII ou multicode d'une chaîne. Les codes de langue sont définis dans l'ISO 639
o	country-code	Spécifie le pays dans lequel la langue est parlée. Les codes de pays sont définis dans l'ISO 3166
m	string	Spécifie la chaîne de texte dans la langue associée

En plus des codes de pays normalisés définis dans l'ISO 3166, le code pays ZZ peut être utilisé pour spécifier une version courte d'une chaîne. Des versions courtes de chaînes sont utiles pour les applications EDD exécutées sur des dispositifs mobiles qui ont une taille d'écran limitée.

EXEMPLE

```

write_protected
"Write Protected"
"|de|Schreibgeschützt"
"|fr|Protégé en écriture"
"|it|Protetto in scrittura"
"|es|Protegido contra escritura"
"|en ZZ|Wr Prot"
"|de ZZ|Geschützt"

```

L'identifiant "write_protected" est l'identifiant qui est référencé dans l'EDD. Les qualificateurs |de|, |fr|, |it|, |es| spécifient le texte pour les langues allemande, française, italienne et espagnole. Les qualificateurs "en ZZ" et "de ZZ" spécifient des versions courtes des chaînes anglaise et allemande. La même approche pourrait également être utilisée pour fournir des versions courtes pour le français, l'italien et l'espagnol.

Annexe A (normative)

Définition formelle d'EDDL

A.1 Préprocesseur EDDL

A.1.1 Structure générale

Objet

Le préprocesseur génère une EDD avant la compilation. Le préprocesseur remplace des parties du texte EDD, insère le contenu d'autres fichiers ou supprime des parties de l'EDD en retirant des sections.

Structure

Le premier caractère d'une ligne de toutes les directives de préprocesseur est un symbole dièse (#). Un espace (caractère SPACE ou TAB) peut apparaître après le signe # initial pour assurer une indentation correcte.

A.1.2 Directives

A.1.2.1 #define

Objet

La directive #define donne un nom significatif à une constante de l'EDD.

Structure

```
#define identifiant token-string
```

Cette forme de la directive #define indique au préprocesseur de remplacer toutes les occurrences suivantes de l'identifiant dans le fichier EDD par token-string.

NOTE L'identifiant n'est pas remplacé s'il apparaît dans un commentaire, dans une chaîne ou dans le cadre d'un identifiant plus long.

```
#define identifiant
```

Cette forme de la directive #define indique au préprocesseur de supprimer toutes les occurrences de l'identifiant du fichier EDD. L'identifiant reste défini et peut être soumis à essai à l'aide de #if défini et des directives #ifdef.

```
#define identifiant (argument, ... ,argument) token-string
```

Cette forme de la directive #define indique au préprocesseur de créer des macros analogues à une fonction. Cette forme accepte une liste d'arguments qui doivent apparaître entre parenthèses et séparés par des virgules.

Si une macro a été définie dans cette forme syntaxique, les instances textuelles subséquentes suivies d'une liste d'arguments constituent un appel de macro. Les arguments réels qui suivent une instance de l'identifiant dans le fichier source sont mis en correspondance avec les paramètres formels correspondants de la définition de macro. Chaque paramètre formel de la chaîne de jetons est remplacé par l'argument réel correspondant. Toutes les macros de l'argument réel sont développées avant que la directive ne remplace le paramètre formel.

Chaque argument de la liste doit être unique.

Aucun espace n'est admis entre l'identifiant et la parenthèse d'ouverture.

Pour les longues directives sur plusieurs lignes source, une barre oblique (\) doit être utilisée avant le caractère NEWLINE.

Aucun espace n'est admis entre le nom et “(“.

A.1.2.2 **#include**

Objet

La directive **#include** indique au préprocesseur de traiter le contenu d'un fichier spécifié comme s'il apparaissait dans le programme source à l'endroit de la directive.

Structure

```
#include "filename"  
#include <filename>
```

Lire le contenu du fichier nommé avec le nom de fichier. Le préprocesseur traite ces données comme si elles faisaient partie intégrante du fichier en cours.

Les deux formes syntaxiques génèrent le remplacement de cette directive par tout le contenu du fichier d'inclusion spécifié. La différence entre les deux formes est l'ordre dans lequel le préprocesseur recherche les fichiers d'inclusion lorsque le chemin d'accès est incomplet.

Aucun jeton supplémentaire n'est admis sur la ligne de directive après le “ ou > final.

A.1.2.3 **#line**

Objet

La directive **#line** permet au préprocesseur de remplacer le numéro de ligne et le nom de fichier stockés en interne du compilateur par un numéro de ligne et un nom de fichier donnés. Le processeur EDD utilise le numéro de ligne et le nom de fichier pour signaler les erreurs trouvées lors du traitement. En règle générale, le numéro de ligne fait référence à la ligne d'entrée en cours, et le nom de fichier au fichier d'entrée en cours. Le numéro de ligne est incrémenté après le traitement de chaque ligne.

Structure

```
#line integer-constant "filename"
```

integer-constant et filename indiquent au préprocesseur de remplacer le numéro de ligne stocké en interne du processeur EDD par un numéro de ligne et un nom de fichier donnés. Integer-constant est interprété comme étant le numéro de la ligne suivante, et est incrémenté après le traitement de chaque ligne.

A.1.2.4 **#if, #elif, #else et #endif**

Objet

La directive **#if**, avec les directives **#elif**, **#else** et **#endif**, contrôle le traitement des parties d'un fichier EDD.

```
#if constant-expression
```

Les lignes suivantes, jusqu'à la directive **#else**, **#elif** ou **#endif**, apparaissent dans la sortie uniquement si constant-expression donne une valeur non nulle. Les opérateurs binaires "&&" et "||" sont légaux dans constant-expression, ainsi que l'opérateur "?:" et les opérateurs unaires "-", "!" et "~".

De plus, l'opérateur unaire défini peut être utilisé dans des expressions constantes particulières, comme l'illustre la syntaxe suivante:

```
defined(identifiant)
```

`defined` `identifiant`

Cette expression constante est considérée comme étant vraie (non nulle) si l'identifiant est réellement défini. Sinon, la condition est fausse. Un identifiant défini comme un texte vide est considéré comme étant défini. La directive définie peut être utilisée dans une directive `#if` et `#elif`, mais nulle part ailleurs.

`#elif` `constant-expression`

Un certain nombre de directives `#elif` peut apparaître entre une directive `#if`, `#ifdef` ou `#ifndef` et une directive `#else` ou `#endif` correspondante. Les lignes qui suivent la directive `#elif` apparaissent dans la sortie uniquement si toutes les conditions suivantes sont vérifiées.

- L'expression constante des directives `#if` précédentes est évaluée à zéro, le nom du `#ifdef` précédent n'est pas défini ou le nom des directives `#ifndef` précédentes a été défini.
- Les expressions constantes de toutes les directives `#elif` d'inversion sont évaluées à zéro.
- L'expression constante en cours est évaluée à une valeur non nulle.

Si l'expression constante est évaluée à une valeur non nulle, ignorer les directives `#elif` et `#else` suivantes jusqu'à la directive `#endif` correspondante. Toutes les expressions constantes admises dans une directive `#if` le sont dans une directive `#elif`.

`#else`

Si toutes les occurrences de l'expression constante sont fausses ou si aucune directive `#elif` n'apparaît, le préprocesseur sélectionne le bloc de texte après `#else`. Si `#else` est ignoré et que toutes les instances de l'expression constante du bloc `#if` sont fausses, aucun bloc de texte n'est sélectionné.

`#endif`

Termine une section de lignes commençant par une des directives conditionnelles `#if`, `#ifdef` ou `#ifndef`. Chaque directive doit comporter un `#endif` correspondant.

NOTE 1 Le préprocesseur reconnaît les paramètres formels des macros dans le corps des directives `#define`, même s'ils apparaissent à l'extérieur des constantes de caractère et des chaînes entre guillemets. Les noms de macro ne sont pas reconnus dans les constantes de caractère ou les chaînes entre guillemets lors d'une analyse régulière. Par conséquent, `#define aaa bbb` ne développe pas `aaa` en LABEL "aaa".

NOTE 2 Les macros ne sont pas développées lors du traitement d'une directive `#define` ou `#undef`. Alors:

`#define aaa bbb`

`#define ccc aaa`

`#undef aaa`

`ccc`

génère `aaa`.

NOTE 3 Les macros sont développées lors de l'analyse qui détermine les paramètres réels vers un autre appel de macro.

Alors:

`#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 et #undef**

Objet

Les directives `#ifdef` et `#ifndef` contrôlent le traitement des parties d'un fichier EDD. Les directives `#ifdef` et `#ifndef` effectuent la même tâche que la directive `#if` si elles sont utilisées avec l'identifiant défini.

`#ifdef identifiant`

Les lignes suivantes, jusqu'à la directive `#else`, `#elif` ou `#endif` correspondante, apparaissent uniquement dans la sortie si l'identifiant a été défini. L'identifiant est spécifié à l'aide d'une directive `#define` ou à l'extérieur de l'EDD, et en l'absence d'une directive `#undef` d'intervention. Aucun jeton supplémentaire n'est admis sur la ligne de directive après le nom.

`#ifndef identifiant`

Les lignes suivantes, jusqu'à la directive `#else`, `#elif` ou `#endif` correspondante, apparaissent dans la sortie uniquement si le nom n'a pas été défini ou si sa définition a été supprimée avec une directive `#undef`. Aucun jeton supplémentaire n'est admis sur la ligne de directive après le nom.

`#undef identifiant`

La directive `#undef` provoque la suppression de la définition de préprocesseur d'un identifiant. Aucun jeton supplémentaire n'est admis sur la ligne de directive après le nom.

A.1.3 **Macros prédéfinies**

A.1.3.1 **Structure générale**

Le préprocesseur doit reconnaître un ensemble spécifié de macros prédéfinies. Il convient d'utiliser ces macros dans l'EDD, lesquelles sont données à titre informatif.

Ces macros ne comportent aucun argument et ne peuvent pas être redéfinies.

A.1.3.2 **Liste des macros prédéfinies**

A.1.3.2.1 **__FILE__**

Objet

`__FILE__` contient le nom du fichier EDD en cours.

Structure

`__FILE__`

`__FILE__` se développe en une chaîne encadrée par des guillemets doubles.

A.1.3.2.2 **__LINE__**

Objet

`__FILE__` contient le numéro de ligne du fichier EDD en cours.

Structure

`__LINE__`

`__LINE__` est une constante entière décimale. Elle peut être modifiée par une directive `#line`.

A.1.4 Caractères NEWLINE

Un caractère NEWLINE termine une constante de caractère ou une chaîne entre guillemets. La barre oblique (\) suivie d'un caractère NEWLINE dans le corps d'une instruction #define est utilisée pour continuer la définition à la ligne suivante.

A.1.5 Commentaires

Un commentaire est une suite de caractères commençant par une combinaison barre oblique/astérisque (/*) et se terminant par une combinaison astérisque/barre oblique (*). Un commentaire peut contenir une combinaison de caractères issus du jeu de caractères représentable, y compris les caractères NEWLINE.

De plus, le commentaire sur une seule ligne est pris en charge s'il est précédé de deux barres obliques (//). Un commentaire commençant par deux barres obliques se termine par le caractère NEWLINE suivant qui n'est pas précédé d'un caractère d'échappement.

A.2 Conventions

A.2.1 Constantes entières

Une constante entière peut être spécifiée en notation binaire, octale, décimale ou hexadécimale. Le Tableau A.1 spécifie les conventions pour chaque type de notation.

Tableau A.1 – Conventions pour les constantes entières

Type de constante entière	Conventions
Binaire	Séquence non vide de chiffres binaires 0 et 1 précédée de 0b ou 0B
Octal	Séquence non vide de chiffres compris entre 0 et 7 commençant par 0
Décimal	Séquence non vide de chiffres décimaux compris entre 0 et 9, ne commençant pas par 0
Hexadécimal	Séquence non vide de chiffres hexadécimaux précédée de 0x ou 0X. Les chiffres hexadécimaux sont composés de chiffres compris entre 0 et 9 et de lettres comprises entre a et f (ou A et F) avec les valeurs 10 à 15, respectivement

A.2.2 Constantes à virgule flottante

Structure lexicale

Une constante à virgule flottante est composée de quatre parties:

- une partie entière, une suite de chiffres décimaux;
- un point décimal (.);
- une partie de fraction, une suite de chiffres décimaux;
- une partie d'exposant, suite éventuellement signée de chiffres décimaux précédée de l'une des lettres e ou E.

Règles

Les règles suivantes s'appliquent à l'aide des constantes à virgule flottante:

- la partie entière ou la fraction peut être ignorée, mais pas les deux;
- le point décimal ou la partie de l'exposant peuvent être ignorés, mais pas les deux.

Exemple: Voici des exemples de constantes à virgule flottante:

.87

48.93

4.8e12

Le calcul de l'expression algébrique doit utiliser le type de données le plus précis.

A.2.3 Littéraux chaînes

Un littéral chaîne est une suite éventuellement vide de caractères placée entre guillemets doubles ("). Les caractères entre guillemets doivent être de type ISO Latin-1 (ISO/CEI 8859-1), à l'exception:

- des guillemets doubles (");
- de la barre oblique (\);
- de la nouvelle ligne.

Grâce aux séquences d'échappement dans les littéraux chaînes, une constante de chaîne peut également contenir des séquences d'échappement représentant un caractère ISO Latin-1. Le Tableau A.2 présente les séquences d'échappement et leur résultat.

Tableau A.2 – Utilisation des séquences d'échappement dans les littéraux chaînes

Code d'échappement	Résultat
'	guillemet simple
"	guillemets doubles
	barre verticale
?	point d'interrogation
\	barre oblique
\a	alerte
\f	alimentation papier
\n	nouvelle ligne
\r	retour chariot
\t	tabulation horizontale
\v	tabulation verticale

A.2.4 Utilisation des codes de langue dans les constantes de chaîne

Un littéral chaîne peut encapsuler toutes les traductions d'une phrase donnée. Le code de langue est spécifié dans l'ISO/CEI 8859-1. Si un littéral chaîne ne contient pas de traduction pour toutes les langues, une langue par défaut doit être utilisée. Le Tableau A.3 présente des exemples de code de langue.

Tableau A.3 – Utilisation des codes de langue dans les littéraux chaînes

Code de langue	Langage
aucun code attribué	langue par défaut, utilisée si une langue demandée n'est pas définie
[cn]	Chinois
[de]	Allemand
[en]	Anglais
[es]	Espagnol
[fr]	Français
[it]	Italien

Code de langue	Langage
[jp]	Japonais
[kr]	Coréen

NOTE L'anglais est recommandé comme langue par défaut.

Exemple Le littéral chaîne suivant spécifie la phrase en anglais "Invalid Selection" et la même phrase en allemand:

"Invalid Selection"

"|de|Unzulässige Auswahl"

A.3 Opérateurs

Le Tableau A.4 contient les opérateurs de l'EDDL.

Tableau A.4 – Opérateurs EDDL

!	!=	%	%=
&	&&	&=	(
)	*	*=	+
++	+=	,	-
--	-=	.	/
/=	:	;	<
<<	<<=	<=	=
==	>	>=	>>
>>=	?	[	]
^	^=	{	
=		}	~
->			

A.4 Mots clés

Le Tableau A.5 contient les mots clés EDDL:

Tableau A.5 – Mots clés EDDL

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
!	COMM	HART	PROGRAMS
%	COMMAND	HEADER	PURPOSE
&	COMMANDS	HELP	READ
(	COMPUTATION	HIDDEN	READ_ONLY
)	CONNECTION	IF	READ_TIMEOUT
*	CONNECTIONS	if	RECORD
+	CONSTANT_UNIT	IGNORE_IN_TEMPORARY_MASTER	RECORDS
,	CONTAINED	IMPORT	REDEFINE
-	continue	INDEX	REDEFINE DOMAIN
.	CORRECTABLE	INFO	REDEFINITIONS
/	CORRECTION	INITIAL_VALUE	REFRESH

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
:	DATA	INPUT	REFRESH_ITEMS
;	DATA_ENTRY_ERROR	int	REFRESHES
<	DATA_ENTRY_WARNING	INTEGER	RELATIONS
=	DATA_EXCHANGE	ITEM_ARRAY	REPLY
>	DATE	ITEM_ARRAY_ITEMS	REQUEST
?	DATE_AND_TIME	ITEM_ARRAYS	RESPONSE_CODES
[	DD_REVISION	ITEMS	return
]	DEFAULT	LABEL	REVIEW
^	default	LAST	ROLE
{	DEFAULT_VALUE	LIKE	ROOT
	DEFINITION	LOAD_TO_APPLICATION	SCALING_FACTOR
}	DELETE	LOAD_TO_DEVICE	SELECT
~	DELETE DOMAIN	LOCAL	SELF
!=	DETAIL	LOCAL_DISPLAY	SELF_CORRECTING
%=	DEVICE	long	SERVICE
&&	DEVICE_REVISION	MANUFACTURER	short
&=	DEVICE_TYPE	MANUFACTURER_EXT	signé
*=	DIAGNOSE	MAX_VALUE	SLOT
++	DIAGNOSTIC	MEMBERS	SOFTWARE
+=	DIALOG	MENU	STATE
--	DIGITAL_INPUT	MENU_ITEMS	STYLE
-=	DIGITAL_OUTPUT	MENUS	SUCCESS
->	DISCRETE_INPUT	METHOD	SUMMARY
/=	DISCRETE_OUTPUT	METHOD_ITEMS	commutateur
<<	DISPLAY_FORMAT	METHODS	TABLEAU
<=	DISPLAY_ITEMS	MIN_VALUE	TEMPS
==	DISPLAY_VALUE	MISC	TIME_VALUE
>=	do	MISC_ERROR	TRANSACTION
>>	DOMAIN	MISC_WARNING	TRANSDUCER
\0	DOMAINS	MODE	TRUE
^=	DOUBLE	MODE_ERROR	TUNE
=	double	MODULE	TYPE
	DURATION	MORE	UNCORRECTABLE
<<=	DYNAMIC	NEXT	UNIT
>>=	EDD_PROFILE	NUMBER	UNIT_ITEMS
ACCESS	EDD_REVISION	NUMBER_OF_ELEMENTS	UNITS
ADD	EDD_VERSION	OBJECT_REFERENCE	unsigned
ALARM	EDIT_DISPLAY	OCTET	UNSIGNED_INTEGER
ANALOG_INPUT	EDIT_DISPLAY_ITEMS	FO	VALIDITY
ANALOG_OUTPUT	EDIT_DISPLAYS	OFFLINE	VARIABLE
APPINSTANCE	EDIT_FORMAT	ONLINE	VARIABLE_LIST
ARGUMENTS	EDIT_ITEMS	OPERATE	VARIABLE_LISTS
ARRAY	ELEMENTS	OPERATION	VARIABLES
ARRAYS	ELSE	OUTPUT	VARIABLE_STATUS

Mot clé EDDL	Mot clé EDDL	Mot clé EDDL	Mot clé EDDL
ASCII	else	PACKED_ASCII	VARIABLE_STATUS_NONE
BAD	ENTRY	PARAMETER_LISTS	VARIABLE_STATUS_INVALID
BIT_ENUMERATED	ENUMERATED	PARAMETERS	VARIABLE_STATUS_NOT_ACCEPTED
BITSTRING	EUC	PARENT	VARIABLE_STATUS_NOT_SUPPORTED
BLOCK	EVENT	PASSWORD	VARIABLE_STATUS_CHANGED
BLOCKS	EVERYTHING	PHYSICAL	VARIABLE_STATUS_LOADED
break	FALSE	POST_EDIT_ACTIONS	VARIABLE_STATUS_INITIAL
CASE	FIRST	POST_READ_ACTIONS	VARIABLE_STATUS_NOT_CONVERTED
case	FLOAT	POST_WRITE_ACTIONS	VISBLE
char	float	PRE_EDIT_ACTIONS	while
CHARACTERISTICS	pour	PRE_READ_ACTIONS	WINDOW
CHILD	FREQUENCY_INPUT	PREV	WRITE
CLASS	FREQUENCY_OUTPUT	PRE_WRITE_ACTIONS	WRITE_AS_ONE
COLLECTION	FUNCTION	PROCESS	WRITE_AS_ONE_ITEMS
COLLECTION_ITEMS	HANDLING	PROCESS_ERROR	WRITE_AS_ONES
COLLECTIONS	HARDWARE	PROGRAM	WRITE_TIMEOUT

A.5 Terminaux

Le texte suivant contient les terminaux admis de la structure lexicale d'EDDL.

```

DEFINE digit          = { 0-9 } .
    bin_digit         = { 0 1 } .
    non_zero_digit    = { 1-9 '-' } .
    oct_digit         = { 0-7 } .
    hex_digit         = { 0-9abcdefABCDEF } .
    letter            = { a-zA-Z } .
    escapes           = { '\"?afnrtv'\" } .
    ISOLatin1char     = - { " } .

/* Integer */
(0b|0B) bin_digit + /* binary */
non_zero_digit digit * /* decimal */
"0" oct_digit * /* octal */
(0x|0X) hex_digit + /* hexadecimal */

/* real */
digit* "." digit+ ((E|e) {+|-}? digit+)?

/* string */
\" ISOLatin1char * \"

/* character */
\' ISOLatin1char \'

/* Identifier */

```

A.6 Syntaxe EDDL letter (letter|digit|_)*Formal

A.6.1 Généralités

La présente annexe contient une syntaxe formelle de l'EDDL utilisant la forme de Backus Naur.

En présence d'éléments facultatifs, chaque élément est marqué d'un /* M */ si l'élément est obligatoire ou d'un /* O */ s'il est facultatif.

A.6.2 Informations d'identification d'une EDD

```

device_description
    = identification definition_list

identification
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_profile
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' manufacturer_ext
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' edd_profile
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' manufacturer_ext
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_profile ',' manufacturer_ext
    = manufacturer ',' device_type ',' device_revision ','
      DD_revision ',' edd_version ',' edd_profile ','
      manufacturer_ext

manufacturer
    = 'MANUFACTURER' Integer
    = 'MANUFACTURER' Identifier

device_type
    = 'DEVICE_TYPE' Integer
    = 'DEVICE_TYPE' Identifier

device_revision
    = 'DEVICE_REVISION' Integer

DD_revision
    = 'DD_REVISION' Integer
    = 'EDD_REVISION' Integer

edd_version
    = 'EDD_VERSION' Integer

edd_profile
    = 'EDD_PROFILE' Integer

manufacturer_ext
    = 'MANUFACTURER_EXT' string

definition_list
    = definition_listR

definition_listR

```

```

= definition
= definition_listR definition

```

```

definition
= item
= imported_description
= like

```

```

item
= axis
= array
= block
= chart
= collection
= command
= connection
= domain
= edit_display
= file
= graph
= list
= item_array
= menu
= method
= program
= record
= refresh_relation
= response_codes_definition
= source
= unit_relation
= variable
= variable_list
= waveform
= wao_relation

```

A.6.3 AXIS

```

axis
= 'AXIS' Identifier '{' axis_attribute_list '}'
= 'AXIS' Identifier '{' '}'

```

```

axis_attribute_list
= axis_attribute_listR

```

```

axis_attribute_listR
= axis_attribute
= axis_attribute_listR axis_attribute

```

```

axis_attribute
= axis_label           /* O */
= axis_min_value       /* O */
= axis_max_value       /* O */
= axis_scaling          /* O */
= axis_help            /* O */
= constant_unit        /* O */

```

```

axis_min_value
= minimum_value

```

```

axis_max_value
= maximum_value

```

```

axis_label
= label

```

```
axis_scaling
    = 'SCALING' axis_scaling_specifier

axis_scaling_specifier
    = axis_scaling_keyword ';'
    = 'IF' '(' expr ')' '{' axis_scaling_specifier '}'
    = 'IF' '(' expr ')' '{' stmt1: axis_scaling_specifier '}'
      'ELSE' '{' stmt2: axis_scaling_specifier '}'
    = 'SELECT' '(' expr ')' '{' axis_scaling_selection_list '}'

axis_scaling_selection_list
    = axis_scaling_selection_listR

axis_scaling_selection_listR
    = axis_scaling_selection
    = axis_scaling_selection_listR axis_scaling_selection

axis_scaling_selection
    = 'CASE' expr ':' axis_scaling_specifier
    = 'DEFAULT' ':' axis_scaling_specifier

axis_scaling_keyword
    = 'LINEAR'
    = 'LOGARITHETIC'

axis_help
    = help
```

A.6.4 BLOCK_A et BLOCK_B

```

block
    = 'BLOCK' Identifier '{' block_attribute_list '}'

block_attribute_list
    = block_attribute
    = block_attribute_list block_attribute

block_attribute
    = block_a_axis_items          /* O */
    = block_a_characteristics      /* M */
    = 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 */
    = block_a_image_items          /* O */
    = block_a_help                 /* O */
    = block_a_item_array_items     /* O */
    = block_a_label                /* M */
    = block_a_list_items           /* O */
    = block_a_local_parameters     /* O */
    = block_a_menu_items           /* O */
    = block_a_method_items         /* O */
    = block_a_parameters           /* M */
    = block_a_parameter_lists      /* O */
    = block_a_refresh_items        /* O */
    = block_a_source_items         /* O */
    = block_a_unit_items           /* O */
    = block_a_wao_items            /* O */
    = block_a_waveform_items       /* O */

    = block_b_number              /* M */
    = block_b_type                /* M */

block_a_help
    = help

block_a_label
    = label

block_a_characteristics
    = 'CHARACTERISTICS' record_reference ';'

block_a_parameters
    = 'PARAMETERS' '{' member_list '}'

member_list
    = member
    = member_list member

block_a_parameter_lists
    = 'PARAMETER_LISTS' '{' 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_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_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_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_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_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_unit_items
    = 'UNIT_ITEMS' '{' unit_items_specifier_list '}'

unit_items_specifier_list
    = unit_items_specifier
    = unit_items_specifier_list ',' unit_items_specifier

unit_items_specifier

```

```

    = unit_reference

block_a_wao_items
    = 'WRITE_AS_ONE_ITEMS' '{' wao_items_specifier_list '}'

wao_items_specifier_list
    = wao_items_specifier
    = wao_items_specifier_list ',' wao_items_specifier

wao_items_specifier
    = wao_reference

block_a_axis_items
    = 'AXIS_ITEMS' '{' axis_items_specifier_list '}'

axis_items_specifier_list
    = axis_items_specifier_listR

axis_items_specifier_listR
    = axis_items_specifier
    = axis_items_specifier_listR ',' 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_listR

chart_items_specifier_listR
    = chart_items_specifier
    = chart_items_specifier_listR ',' chart_items_specifier

chart_items_specifier
    = chart_reference

block_a_file_items
    = 'FILE_ITEMS' '{' file_items_specifier_list '}'

file_items_specifier_list
    = file_items_specifier_listR

file_items_specifier_listR
    = file_items_specifier
    = file_items_specifier_listR ',' 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_listR

graph_items_specifier_listR
    = graph_items_specifier
    = graph_items_specifier_listR ',' 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_listR

grid_items_specifier_listR
    = grid_items_specifier
    = grid_items_specifier_listR ',' 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_listR

image_items_specifier_listR
    = image_items_specifier
    = image_items_specifier_listR ',' image_items_specifier

image_items_specifier
    = image_reference

block_a_list_items
    = 'LIST_ITEMS' '{' list_items_specifier_list '}'

list_items_specifier_list
    = list_items_specifier_listR

list_items_specifier_listR
    = list_items_specifier
    = list_items_specifier_listR ',' list_items_specifier

list_items_specifier
    = list_reference

block_a_source_items
    = 'SOURCE_ITEMS' '{' source_items_specifier_list '}'

source_items_specifier_list
    = source_items_specifier_listR

source_items_specifier_listR
    = source_items_specifier
    = source_items_specifier_listR ',' source_items_specifier

source_items_specifier
    = source_reference

block_a_waveform_items
    = 'WAVEFORM_ITEMS' '{' waveform_items_specifier_list '}'

waveform_items_specifier_list
    = waveform_items_specifier_listR

waveform_items_specifier_listR
    = waveform_items_specifier
    = waveform_items_specifier_listR ',' waveform_items_specifier
```



```

waveform_items_specifier
    = waveform_reference

block_a_local_parameters
    = 'LOCAL_PARAMETERS' '{ member_list }'

block_b_number
    = 'NUMBER' expr ';'

block_b_type
    = 'TYPE' 'PHYSICAL' ';';
    = 'TYPE' 'TRANSDUCER' ';';
    = 'TYPE' 'FUNCTION' ';';

```

A.6.5 CHART

```

chart
    = 'CHART' Identifier '{ chart_attribute_list }'

chart_attribute_list
    = chart_attribute_listR

chart_attribute_listR
    = chart_attribute
    = chart_attribute_listR chart_attribute

chart_attribute
    = chart_label /* O */
    = chart_help /* O */
    = chart_members /* M */
    = chart_height /* O */
    = chart_width /* O */
    = chart_type /* O */
    = chart_cycle_time /* O */
    = chart_length /* O */

chart_label
    = label

chart_help
    = help

chart_members
    = members

chart_height
    = 'HEIGHT' display_size_specifier

chart_width
    = 'WIDTH' display_size_specifier

chart_type
    = 'TYPE' chart_type_specifier

chart_type_specifier
    = chart_type_keyword ';'
    = 'IF' '(' expr ')' '{ chart_type_specifier }'
    = 'IF' '(' expr ')' '{ stmt1: chart_type_specifier }'
    'ELSE' '{ stmt2: chart_type_specifier }'
    = 'SELECT' '(' expr ')' '{ chart_type_selection_list }'

chart_type_selection_list
    = chart_type_selection_listR

```

```

chart_type_selection_listR
    = chart_type_selection
    = chart_type_selection_listR 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'

chart_cycle_time
    = 'CYCLE_TIME' expr_specifier

chart_length
    = 'LENGTH' expr_specifier

```

A.6.6 COLLECTION

```

collection
    = 'COLLECTION' Identifier '{' collection_attribute_list '}'
    = 'COLLECTION' 'OF' collection_item_type Identifier '{'
      collection_attribute_list '}'

collection_item_type
    = item_type

item_type
    = 'ARRAY'
    = 'AXIS'
    = 'BLOCK'
    = 'CHART'
    = 'COLLECTION' 'OF' item_type
    = 'COMMAND'
    = 'CONNECTION'
    = 'DOMAIN'
    = 'EDIT_DISPLAY'
    = 'FILE'
    = 'GRAPH'
    = 'ITEM_ARRAY' 'OF' item_type
    = 'LIST'
    = 'MENU'
    = 'METHOD'
    = 'PROGRAM'
    = 'RECORD'
    = 'REFRESH'
    = 'RESPONSE_CODES'
    = 'SOURCE'
    = 'UNIT'
    = 'VARIABLE'
    = 'VARIABLE_LIST'
    = 'WAVEFORM'
    = 'WRITE_AS_ONE'

collection_attribute_list
    = collection_attribute
    = collection_attribute_list collection_attribute

```

```
collection_attribute
    = help                /* O */
    = label               /* O */
    = members             /* M */
```

A.6.7 COMMAND

```
command
    = 'COMMAND' Identifier '{' command_attribute_list '}'
    = 'COMMAND' Identifier '{' '}'
```

```
command_attribute_list
    = command_attribute
    = command_attribute_list command_attribute
```

```
command_attribute
    = command_header      /* O */
    = command_slot        /* O */
    = command_index       /* O */
    = command_block       /* O */
    = command_number      /* O */
    = command_operation    /* M */
    = command_transaction /* M */
    = command_connection  /* O */
    = command_module      /* O */
    = command_response_codes /* O */
```

```
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
```

```
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_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_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' number_specifier

number_specifier:
    = number_items ';'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    = 'ELSE' '{' number_specifier '}'
    = 'SELECT' '(' expr ')' '{' number_selection_list '}'

number_items:
    = Integer
    = variable_reference

number_selection_list
    = number_selection
    = number_selection_list number_selection

number_selection

```

```

= 'CASE' expr ':' number_specifier
= 'DEFAULT' ':' number_specifier

command_operation
= 'OPERATION' operation_specifier

operation_specifier
= operation ';'
= 'IF' '(' expr ')' '{' operation_specifier '}'
= 'IF' '(' expr ')' '{' operation_specifier '}'
  'ELSE' '{' operation_specifier '}'
= 'SELECT' '(' expr ')' '{' operation_selection_list '}'

operation_selection_list
= operation_selection
= operation_selection_list operation_selection

operation_selection
= 'CASE' expr ':' operation_specifier
= 'DEFAULT' ':' operation_specifier

operation
= 'READ'
= 'WRITE'
= 'COMMAND'
= 'DATA_EXCHANGE'
= string

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 */

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
= data_items
= data_items_list ',' data_items

data_items
= Integer
= variable_reference
= variable_reference '<' Integer '>'
= variable_reference '(' data_items_qualifiers ')'
= variable_reference '<' Integer '>' '(' data_items_qualifiers ')'

```

```

data_items_qualifiers
    = data_items_qualifiers_wrap

data_items_qualifiers_wrap
    = data_items_qualifier
    = data_items_qualifiers_wrap ',' data_items_qualifier

data_items_qualifier
    = 'INDEX'
    = 'INFO'

command_connection
    = 'CONNECTION' connection_specifier

connection_specifier
    = connection_reference ';'
    = 'IF' '(' expr ')' '{' connection_specifier '}'
    = 'IF' '(' expr ')' '{' connection_specifier '}'
    'ELSE' '{' connection_specifier '}'
    = 'SELECT' '(' expr ')' '{' connection_selection_list '}'

connection_selection_list
    = connection_selection
    = connection_selection_list connection_selection

connection_selection
    = 'CASE' expr ':' connection_specifier
    = 'DEFAULT' ':' connection_specifier

command_module
    = 'MODULE' module_specifier

module_specifier
    = Identifier ';'
    = 'IF' '(' expr ')' '{' module_specifier '}'
    = 'IF' '(' expr ')' '{' module_specifier '}'
    'ELSE' '{' module_specifier '}'
    = 'SELECT' '(' expr ')' '{' module_selection_list '}'

module_selection_list
    = module_selection
    = module_selection_list module_selection

module_selection
    = 'CASE' expr ':' module_specifier
    = 'DEFAULT' ':' module_specifier

command_response_codes
    = response_codes

```

A.6.8 CONNECTION

```

connection
    = 'CONNECTION' Identifier '{' connection_attribute_list '}'

connection_attribute_list
    = connection_attribute
    = connection_attribute_list connection_attribute

connection_attribute
    = appinstance

```

/* M */

```
appinstance
    = 'APPINSTANCE' Integer ';'

```

A.6.9 DOMAIN

```
domain
    = 'DOMAIN' Identifier '{' domain_attribute_list '}'
    = 'DOMAIN' Identifier '{' '}'

domain_attribute_list
    = domain_attribute
    = domain_attribute_list domain_attribute

domain_attribute
    = handling /* O */
    = response_codes /* O */

```

A.6.10 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
    = display_items /* O */
    = edit_items /* M */
    = label /* M */
    = post_edit_actions /* O */
    = pre_edit_actions /* O */

edit_items
    = 'EDIT_ITEMS' '{' edit_items_specifier_list '}'

edit_items_specifier_list
    = edit_items_specifier
    = edit_items_specifier_list edit_items_specifier

edit_items_specifier
    = edit_item_list
    = 'IF' '(' expr ')' '{' edit_items_specifier_list '}'
    = 'IF' '(' expr ')' '{' edit_items_specifier_list '}'
    = 'ELSE' '(' edit_items_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' edit_items_selection_list '}'

edit_items_selection_list
    = edit_items_selection
    = edit_items_selection_list edit_items_selection

edit_items_selection
    = 'CASE' expr ':' edit_items_specifier_list
    = 'DEFAULT' ':' edit_items_specifier_list

edit_item_list
    = edit_item
    = edit_item_list ',' edit_item

edit_item
    = reference_without_call

display_items
    = 'DISPLAY_ITEMS' '{' display_items_specifier_list '}'

```

```

display_items_specifier_list
    = display_items_specifier
    = display_items_specifier_list display_items_specifier

display_items_specifier
    = display_item_list
    = 'IF' '(' expr ')' '{' display_items_specifier_list '}'
    = 'IF' '(' expr ')' '{' display_items_specifier_list '}'
    'ELSE' '{' display_items_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' display_items_selection_list '}'

display_items_selection_list
    = display_items_selection
    = display_items_selection_list display_items_selection

display_items_selection
    = 'CASE' expr ':' display_items_specifier_list
    = 'DEFAULT' ':' display_items_specifier_list

display_item_list
    = display_item
    = display_item_list ',' display_item

display_item
    = variable_reference

```

A.6.11 FILE

```

file
    = 'FILE' Identifier '{' file_attribute_list '}'

file_attribute_list
    = file_attribute_listR

file_attribute_listR
    = file_attribute
    = file_attribute_listR file_attribute

file_attribute
    = file_label                /* O */
    = file_help                 /* O */
    = file_members              /* M */

file_label
    = label

file_help
    = help

file_members
    = members

```

A.6.12 GRAPH

```

graph
    = 'GRAPH' Identifier '{' graph_attribute_list '}'

graph_attribute_list
    = graph_attribute_listR

graph_attribute_listR
    = graph_attribute
    = graph_attribute_listR graph_attribute

```



```

graph_attribute
    = graph_label                /* O */
    = graph_help                 /* O */
    = graph_members              /* M */
    = graph_x_axis               /* O */
    = graph_height               /* O */
    = graph_width                /* O */
    = graph_cycle_time           /* O */

graph_label
    = label

graph_help
    = help

graph_members
    = members

graph_x_axis
    = 'X_AXIS' axis_specifier

graph_height
    = 'HEIGHT' display_size_specifier

graph_width
    = 'WIDTH' display_size_specifier

display_size_specifier
    = display_size_keyword ';'
    = 'IF' '(' expr ')' '{' display_size_specifier '}'
    = 'IF' '(' expr ')' '{' stmt1: display_size_specifier '}'
      'ELSE' '{' stmt2: display_size_specifier '}'
    = 'SELECT' '(' expr ')' '{' display_size_selection_list '}'

display_size_selection_list
    = display_size_selection_listR

display_size_selection_listR
    = display_size_selection
    = display_size_selection_listR display_size_selection

display_size_selection
    = 'CASE' expr ':' display_size_specifier
    = 'DEFAULT' ':' display_size_specifier

display_size_keyword
    = 'XX_SMALL'
    = 'X_SMALL'
    = 'SMALL'
    = 'MEDIUM'
    = 'XX_LARGE'
    = 'X_LARGE'
    = 'LARGE'

graph_cycle_time
    = 'CYCLE_TIME' expr_specifier

```

A.6.13 GRID

```

grid
    = 'GRID' Identifier '{' grid_attribute_list '}'

```

```

grid_attribute_list
    = grid_attribute_listR

grid_attribute_listR
    = grid_attribute
    = grid_attribute_listR grid_attribute

grid_attribute
    = grid_handling
    = grid_height
    = grid_help
    = grid_label
    = grid_orientation
    = grid_validity
    = grid_vectors
    = grid_width
                                /* O */
                                /* O */
                                /* O */
                                /* O */
                                /* O */
                                /* O */
                                /* M */
                                /* O */

grid_handling
    = 'HANDLING' handling_specifier

grid_height
    = 'HEIGHT' display_size_specifier

grid_help
    = help

grid_label
    = label

grid_orientation
    = 'ORIENTATION' orientation_specifier ';'

orientation_specifier
    = 'HORIZONTAL'
    = 'VERTICAL'

grid_vectors
    = 'VECTORS' '{' vector_specifier_list '}'

vector_specifier_list
    = vector_specifier_listR

vector_specifier_listR
    = vector_specifier
    = vector_specifier_listR ',' vector_specifier

vector_specifier
    = vector
    = 'IF' '(' expr ')' '{' vector_specifier_list '}'
    = 'IF' '(' expr ')' '{' stmt1: vector_specifier_list '}'
    'ELSE' '{' stmt2: 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_listR

vector_values_listR
    = vector_values
    = vector_values_listR ',' vector_values

vector_values
    = reference_without_call
    = Integer
    = Real
    = string

grid_width
    = 'WIDTH' display_size_specifier
```

A.6.14 IMAGE

```
image
    = 'IMAGE' Identifier '{' image_attribute_list '}'

image_attribute_list
    = image_attribute_listR

image_attribute_listR
    = image_attribute
    = image_attribute_listR image_attribute

image_attribute
    = image_label /* O */
    = image_help /* O */
    = image_link /* O */
    = image_path /* M */
    = image_validity /* O */

image_label
    = label

image_help
    = help

image_link
    = 'LINK' link_specifier

link_specifier
    = reference_no_call ';'
    = 'IF' '(' expr ')' '{' link_specifier '}'
    = 'IF' '(' expr ')' '{' stmt1: link_specifier '}'
    'ELSE' '{' stmt2: link_specifier '}'
    = 'SELECT' '(' expr ')' '{' link_selection_list '}'

link_selection_list
    = link_selection_listR

link_selection_listR
    = link_selection
    = link_selection_listR link_selection

link_selection
    = 'CASE' expr ':' link_specifier
    = 'DEFAULT' ':' link_specifier
```

```

image_path
    = 'PATH' string_specifier

image_validity
    = validity

```

A.6.15 LIST

```

list
    = 'LIST' Identifier '{' list_attribute_list '}'

list_attribute_list
    = list_attribute_listR

list_attribute_listR
    = list_attribute
    = list_attribute_listR list_attribute

list_attribute
    = list_label                /* O */
    = list_help                 /* O */
    = list_capacity             /* O */
    = list_count                /* O */
    = list_type                 /* M */

list_label
    = label

list_help
    = help

list_capacity
    = 'CAPACITY' expr_specifier ';'

list_count
    = 'COUNT' expr_specifier ';'

list_type
    = 'TYPE' list_type_specifier ';'

list_type_specifier
    = reference_without_call

```

A.6.16 IMPORT

```

imported_description
    = 'IMPORT' identification '{' imports '}'
    = 'IMPORT' identification '{' imports redefinitions '}'
    = 'IMPORT' '"' filename '"'
    = 'IMPORT' '<' filename '>'

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'
= 'CONNECTIONS'
= 'DOMAINS'
= 'EDIT_DISPLAYS'
= 'FILES'
= 'GRAPHS'
= 'ITEM_ARRAYS'
= 'LISTS'
= 'MENUS'
= 'METHODS'
= 'PROGRAMS'
= 'RECORDS'
= 'REFRESHES'
= 'RELATIONS'
= 'RESPONSE_CODES'
= 'SOURCES'
= 'UNITS'
= 'VARIABLES'
= 'VARIABLE_LISTS'
= 'WRITE_AS_ONES'
= 'WAVEFORMS'

redefinitions
= 'REDEFINITIONS' '{' redefinition_list '}'

redefinition_list
= redefinition
= redefinition_list redefinition

redefinition
= array_redefinition
= axis_redefinition
= block_redefinition
= chart_redefinition
= collection_redefinition
= command_redefinition
= connection_redefinition
= domain_redefinition
= edit_display_redefinition
= file_definition
= graph_definition
= item_array_redefinition
= list_redefinition
= menu_redefinition
= method_redefinition
= program_redefinition
= record_redefinition
= refresh_relation_redefinition
= response_codes_definition_redefinition
= source_redefinition

```

```

= unit_relation_redefinition
= variable_redefinition
= variable_list_redefinition
= wao_relation_redefinition
= waveform_redefinition

```

A.6.17 LIKE

like

```

= Identifier 'LIKE' Identifier
  '{' attribute_redefinition_list '}'
= Identifier 'LIKE' 'ARRAY' Identifier
  '{' array_attribute_redefinition_list '}'
= Identifier 'LIKE' 'BLOCK' Identifier
  '{' block_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COLLECTION' 'OF' item_type Identifier
  '{' collection_attribute_redefinition_list '}'
= Identifier 'LIKE' 'COMMAND' Identifier
  '{' command_attribute_redefinition_list '}'
= Identifier 'LIKE' 'CONNECTION' Identifier
  '{' connection_attribute_redefinition_list '}'
= Identifier 'LIKE' 'DOMAIN' Identifier
  '{' domain_attribute_redefinition_list '}'
= Identifier 'LIKE' 'EDIT_DISPLAY' Identifier
  '{' edit_display_attribute_redefinition_list '}'
= Identifier 'LIKE' 'ITEM_ARRAY' 'OF' item_type
  Identifier
  '{' item_array_attribute_redefinition_list '}'
= Identifier 'LIKE' 'MENU' Identifier
  '{' menu_attribute_redefinition_list '}'
= Identifier 'LIKE' 'METHOD' Identifier
  '{' method_attribute_redefinition_list '}'
= Identifier 'LIKE' 'PROGRAM' Identifier
  '{' program_attribute_redefinition_list '}'
= Identifier 'LIKE' 'RECORD' Identifier
  '{' record_attribute_redefinition_list '}'
= Identifier 'LIKE' 'REFRESH' Identifier
  '{' '}'
= Identifier 'LIKE' 'RESPONSE_CODES' Identifier
  '{' response_code_redefinition_list '}'
= Identifier 'LIKE' 'UNIT' Identifier
  '{' '}'
= Identifier 'LIKE' 'VARIABLE' Identifier
  '{' variable_attribute_redefinition_list '}'
= Identifier 'LIKE' 'VARIABLE_LIST' Identifier
  '{' variable_list_attribute_redefinition_list '}'
= Identifier 'LIKE' 'WRITE_AS_ONE' Identifier
  '{' '}'
= Identifier 'LIKE' 'AXIS' Identifier
  '{' '}'
= Identifier 'LIKE' 'CHART' Identifier
  '{' '}'
= Identifier 'LIKE' 'GRAPH' Identifier
  '{' '}'
= Identifier 'LIKE' 'FILE' Identifier
  '{' '}'
= Identifier 'LIKE' 'LIST' Identifier
  '{' '}'
= Identifier 'LIKE' 'SOURCE' Identifier
  '{' '}'
= Identifier 'LIKE' 'WAVEFORM' Identifier
  '{' '}'

```

attribute_redefinition_list

```
= attribute_redefinition
= attribute_redefinition_list attribute_redefinition
```

```
attribute_redefinition
= appinstance_redefinition
= arguments_redefinition
= array_type_redefinition
= block_a_characteristics_redefinition
= block_a_collection_item_redefinition
= block_a_edit_disp_item_redefinition
= block_a_item_array_item_redefinition
= block_a_menu_item_redefinition
= block_a_method_item_redefinition
= block_a_parameters_redefinition
= block_a_parameter_lists_redefinition
= block_a_refresh_item_redefinition
= block_a_unit_item_redefinition
= block_a_wao_item_redefinition
= block_b_number_redefinition
= block_b_type_redefinition
= command_slot_redefinition
= command_index_redefinition
= command_block_redefinition
= command_operation_redefinition
= command_transaction_redefinition
= command_connection_redefinition
= command_module_redefinition
= display_items_redefinition
= edit_items_redefinition
= elements_redefinition
= handling_redefinition
= help_redefinition
= members_redefinition
= menu_access_redefinition
= menu_entry_redefinition
= menu_items_redefinition
= menu_purpose_redefinition
= menu_style_redefinition
= method_definition_redefinition
= number_of_elements_redefinition
= optional_label_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
= read_timeout_redefinition
= response_code_redefinition
= response_codes_reference_redefinition
= type_redefinition
= validity_redefinition
= variable_class_redefinition
= write_timeout_redefinition
```

A.6.18 MENU

```
menu
= 'MENU' Identifier '{' menu_attribute_list '}'

menu_attribute_list
= menu_attribute
= menu_attribute_list menu_attribute
```

```

menu_attribute
    = help                      /* O */
    = label                    /* M */
    = menu_access              /* O */
    = menu_entry              /* O */
    = menu_items              /* M */
    = menu_purpose              /* O */
    = menu_role               /* O */
    = menu_style              /* O */
    = validity                /* O */
    = pre_edit_actions        /* O */
    = pre_read_actions        /* O */
    = pre_write_actions       /* O */
    = post_edit_actions       /* O */
    = post_read_actions       /* O */
    = post_write_actions      /* O */

menu_access
    = 'ACCESS' 'ONLINE' ';'
    = 'ACCESS' 'OFFLINE' ';'

menu_entry
    = 'ENTRY' menu_entry_specifier

menu_entry_specifier
    = menu_entry_item ';'
    = 'IF' '(' expr ')' '{' menu_entry_specifier '}'
    = 'IF' '(' expr ')' '{' menu_entry_specifier '}'
    = 'ELSE' '{' menu_entry_specifier '}'
    = 'SELECT' '(' expr ')' '{' menu_entry_selection_list '}'

menu_entry_selection_list
    = menu_entry_selection
    = menu_entry_selection_list menu_entry_selection

menu_entry_selection
    = 'CASE' expr ':' menu_entry_specifier
    = 'DEFAULT' ':' menu_entry_specifier

menu_entry_item
    = 'ROOT'

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_list

menu_item_selection_list
    = menu_item_selection

```



```

    = menu_item_selection_list menu_item_selection

menu_item_selection
    = 'CASE' expr ':' menu_item_specifier_list
    = 'DEFAULT' ':' menu_item_specifier_list

menu_item_list
    = menu_item
    = menu_item_list ',' menu_item

menu_item
    = reference_without_call
    = reference_without_call menu_item_args
    = 'SEPARATOR'
    = 'COLUMNBREAK'
    = 'ROWBREAK'

menu_item_args
    = '(' argument_list ')'
    = '(' 'REVIEW' ')'
    = '(' variable_qualifier_list ')'
    = '(' image_qualifier_list ')'

variable_qualifier_list
    = variable_qualifier
    = variable_qualifier_list ',' variable_qualifier

variable_qualifier
    = 'DISPLAY_VALUE'
    = 'READ_ONLY'
    = 'HIDDEN'
    = 'NO_LABEL'
    = 'NO_UNIT'

image_qualifier_list
    = image_qualifier
    = image_qualifier_list ',' image_qualifier

image_qualifier
    = 'INLINE'

menu_purpose
    = 'PURPOSE' menu_purpose_specifier

menu_purpose_specifier
    = menu_purpose_list ';'
    = 'IF' '(' expr ')' '{' menu_purpose_specifier '}'
    = 'IF' '(' expr ')' '{' menu_purpose_specifier '}'
    = 'ELSE' '{' menu_purpose_specifier '}'
    = 'SELECT' '(' expr ')' '{' menu_purpose_selection_list '}'

menu_purpose_selection_list
    = menu_purpose_selection
    = menu_purpose_selection_list menu_purpose_selection

menu_purpose_selection
    = 'CASE' expr ':' menu_purpose_specifier
    = 'DEFAULT' ':' menu_purpose_specifier

menu_purpose_list
    = menu_purpose_item
    = menu_purpose_list '&' menu_purpose_item

```

```

menu_purpose_item
    = 'CATALOG'
    = 'DIAGNOSE'
    = 'LOAD_TO_APPLICATION'
    = 'LOAD_TO_DEVICE'
    = 'MENU'
    = 'PROCESS_VALUE'
    = 'SIMULATION'
    = 'TABLE'
    = string

menu_role
    = 'ROLE' menu_role_specifier

menu_role_specifier
    = menu_role_list ';'
    = 'IF' '(' expr ')' '{' menu_role_specifier '}'
    = 'IF' '(' expr ')' '{' menu_role_specifier '}'
    'ELSE' '{' menu_role_specifier '}'
    = 'SELECT' '(' expr ')' '{' menu_role_selection_list '}'

menu_role_selection_list
    = menu_role_selection
    = menu_role_selection_list menu_role_selection

menu_role_selection
    = 'CASE' expr ':' menu_role_specifier
    = 'DEFAULT' ':' menu_role_specifier

menu_role_list
    = menu_role_item
    = menu_role_list '&' menu_role_item

menu_role_item
    = string

menu_style
    = 'STYLE' menu_style_specifier

menu_style_specifier
    = menu_style_item ';'
    = 'IF' '(' expr ')' '{' menu_style_specifier '}'
    = 'IF' '(' expr ')' '{' menu_style_specifier '}'
    'ELSE' '{' menu_style_specifier '}'
    = 'SELECT' '(' expr ')' '{' menu_style_selection_list '}'

menu_style_selection_list
    = menu_style_selection
    = menu_style_selection_list menu_style_selection

menu_style_selection
    = 'CASE' expr ':' menu_style_specifier
    = 'DEFAULT' ':' menu_style_specifier

menu_style_item    = 'DIALOG'
    = 'GROUP'
    = 'MENU'
    = 'PAGE'
    = 'TABLE'
    = 'WINDOW'
    = string

```

A.6.19 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
    = 'char'
    = 'unsigned char'
    = 'int'
    = 'unsigned int'
    = 'long'
    = 'unsigned long'
    = 'float'
    = 'double'
    = 'DD_STRING'
    = 'DD_ITEM'

method_attribute_list
    = method_attribute
    = method_attribute_list method_attribute

method_attribute
    = help /* O */
    = label /* M */
    = method_access /* O */
    = method_definition /* M */
    = method_type /* O */
    = validity /* M */
    = variable_class /* M */

method_access
    = 'ACCESS' 'OFFLINE' ';'
    = 'ACCESS' 'ONLINE' ';'

method_definition
    = 'DEFINITION' c_compound_statement

method_type
    = 'TYPE' method_return_type

method_return_type
    = 'char'
    = 'unsigned char'
    = 'int'
    = 'unsigned int'
    = 'long'
    = 'unsigned long'
    = 'float'
    = 'double'
    = 'DD_STRING'

```

A.6.20 PROGRAM

```

program
    = 'PROGRAM' Identifier '{' program_attribute_list '}'
    = 'PROGRAM' Identifier '{' '}'

program_attribute_list
    = program_attribute
    = program_attribute_list program_attribute

program_attribute
    = arguments                      /* O */
    = response_codes                 /* O */

arguments
    = 'ARGUMENTS' '{' '}'
    = 'ARGUMENTS' '{' arguments_specifier_list '}'

arguments_specifier_list
    = arguments_specifier
    = arguments_specifier_list arguments_specifier

arguments_specifier
    = argument_list
    = 'IF' '(' expr ')' '{' arguments_specifier_list '}'
    = 'IF' '(' expr ')' '{' arguments_specifier_list '}'
    = 'ELSE' '{' arguments_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' arguments_selection_list '}'

arguments_selection_list
    = arguments_selection
    = arguments_selection_list arguments_selection

arguments_selection
    = 'CASE' expr ':' arguments_specifier_list
    = 'DEFAULT' ':' arguments_specifier_list

argument_list
    = argument
    = argument_list ',' argument

argument
    = Integer
    = variable_reference

```

A.6.21 RECORDS

```

record
    = 'RECORD' Identifier '{' record_attribute_list '}'

record_attribute_list
    = record_attribute
    = record_attribute_list record_attribute

record_attribute
    = help                          /* O */
    = label                         /* M */
    = members                       /* M */
    = response_codes                /* O */

```

A.6.22 REFERENCE_ARRAY

Reference_Array comporte deux mots clés disponibles, "ITEM_ARRAY" (première option) et "ARRAY" (deuxième option) qui peuvent être sélectionnés par un profil. Le mot clé ARRAY ne doit pas être sélectionné si value_array est utilisé.

NOTE Si une application EDD prend en charge plusieurs profils, le mot clé ARRAY peut être également utilisé à la place d'ITEM_ARRAY. Dans ce cas, la sélection sémantique est assurée par les attributs.

```

item_array
= 'ITEM_ARRAY' 'OF' item_array_attribute_type Identifier '{'
item_array_attribute_list '}'
= 'ARRAY' 'OF' item_array_attribute_type Identifier '{'
item_array_attribute_list '}'

item_array_attribute_type
= item_array_type

item_array_type
= 'ARRAY'
= 'BLOCK'
= 'COLLECTION' 'OF' item_array_type
= 'COMMAND'
= 'CONNECTION'
= 'DOMAIN'
= 'ITEM_ARRAY' 'OF' item_array_type
= 'MENU'
= 'METHOD'
= 'PROGRAM'
= 'RECORD'
= 'REFRESH'
= 'RESPONSE_CODES'
= 'UNIT'
= 'VARIABLE'
= 'VARIABLE_LIST'
= 'WRITE_AS_ONE'

item_array_attribute_list
= item_array_attribute
= item_array_attribute_list item_array_attribute

item_array_attribute
= elements /* M */
= help /* O */
= label /* O */

elements
= 'ELEMENTS' '{' elements_specifier_list '}'

elements_specifier_list
= elements_specifier
= elements_specifier_list elements_specifier

elements_specifier
= element
= 'IF' '(' expr ')' '{' elements_specifier_list '}'
= 'IF' '(' expr ')' '{' elements_specifier_list '}'
= 'ELSE' '{' elements_specifier_list '}'
= 'SELECT' '(' expr ')' '{' elements_selection_list '}'

elements_selection_list
= elements_selection
= elements_selection_list elements_selection

```

```

elements_selection
    = 'CASE' expr ':' elements_specifier_list
    = 'DEFAULT' ':' elements_specifier_list

element
    = Integer ',' reference_without_call ';'
    = Integer ',' reference_without_call ',' description_string ';'
    = Integer ',' reference_without_call ',' description_string ','
help_string ';'

```

A.6.23 Relations

```

refresh_relation
    = 'REFRESH' Identifier
      '{' refresh_relation_left ':' refresh_relation_right '}'

refresh_relation_left
    = variable_reference_specifier_list

refresh_relation_right
    = variable_reference_specifier_list

unit_relation
    = 'UNIT' Identifier
      '{' unit_relation_left ':' unit_relation_right '}'

unit_relation_left
    = variable_reference_specifier

unit_relation_right
    = variable_axis_reference_specifier_list

wao_relation
    = 'WRITE_AS_ONE' Identifier '{' wao_specifier '}'

wao_specifier
    = variable_reference_specifier_list

variable_reference_specifier_list
    = variable_reference_specifier
    = variable_reference_specifier_list variable_reference_specifier
    = variable_reference_specifier_list ',' variable_reference_specifier

variable_reference_specifier
    = variable_reference
    = 'IF' '(' expr ')' '{' variable_reference_specifier '}'
    = 'IF' '(' expr ')' '{' variable_reference_specifier '}'
    = 'ELSE' '{' variable_reference_specifier '}'
    = 'SELECT' '(' expr ')' '{' variable_reference_selection_list '}'

variable_reference_selection_list
    = variable_reference_selection
    = variable_reference_selection_list variable_reference_selection

variable_reference_selection
    = 'CASE' expr ':' variable_reference_specifier
    = 'DEFAULT' ':' variable_reference_specifier

variable_axis_reference_selection_list
    = variable_axis_reference_selection_listR

variable_axis_reference_selection_listR

```

```

        = variable_axis_reference_selection
        = variable_axis_reference_selection_listR
variable_axis_reference_selection

variable_axis_reference_selection
    = 'CASE' expr ':' variable_axis_reference_specifier
    = 'DEFAULT' ':' variable_axis_reference_specifier

variable_axis_reference_specifier
    = variable_reference
    = axis_reference
    = 'IF' '(' expr ')' '{' variable_axis_reference_specifier '}'
    = 'IF' '(' expr ')' '{' stmt1: variable_axis_reference_specifier '}'
    'ELSE' '{' stmt2: variable_axis_reference_specifier '}'
    = 'SELECT' '(' expr ')' '{' variable_axis_reference_selection_list '}'

```

A.6.24 RESPONSE_CODES

```

response_codes_definition
    = 'RESPONSE_CODES' Identifier '{' response_codes_attribute_list
    '}'

response_codes_attribute_list
    = response_codes_specifier_list

response_codes_specifier_list
    = response_codes_specifier
    = response_codes_specifier_list response_codes_specifier

response_codes_specifier
    = response_code
    = 'IF' '(' expr ')' '{' response_codes_specifier_list '}'
    = 'IF' '(' expr ')' '{' response_codes_specifier_list '}'
    'ELSE' '{' response_codes_specifier_list '}'
    = 'SELECT' '(' expr ')' '{' response_codes_selection_list '}'

response_codes_selection_list
    = response_codes_selection
    = response_codes_selection_list response_codes_selection

response_codes_selection
    = 'CASE' expr ':' response_codes_specifier_list
    = 'DEFAULT' ':' response_codes_specifier_list

response_code
    = Integer ',' response_code_type ',' description_string ';'
    = Integer ',' response_code_type ',' description_string ','
    help_string ';'

response_code_type
    = 'SUCCESS'
    = 'MISC_WARNING'
    = 'DATA_ENTRY_WARNING'
    = 'DATA_ENTRY_ERROR'
    = 'MODE_ERROR'
    = 'PROCESS_ERROR'
    = 'MISC_ERROR'

```

A.6.25 SOURCE

```

source
    = 'SOURCE' Identifier '{' source_attribute_list '}'

```

```

source_attribute_list
    = source_attribute_listR

source_attribute_listR
    = source_attribute
    = source_attribute_listR source_attribute

source_attribute
    = source_help                /* O */
    = source_label                /* O */
    = source_emphasis            /* O */
    = source_line_color          /* O */
    = source_line_type           /* O */
    = source_y_axis              /* O */
    = source_members             /* M */
    = source_init_actions        /* O */
    = source_refresh_actions     /* O */
    = source_exit_actions        /* O */

source_help
    = help

source_label
    = label

source_emphasis
    = 'EMPHASIS' boolean_specifier

source_line_color
    = 'LINE_COLOR' line_color_specifier

source_line_type
    = 'LINE_TYPE' line_type_specifier ';'

source_y_axis
    = 'Y_AXIS' axis_specifier

source_members
    = members

source_init_actions
    = 'INIT_ACTIONS' '{' actions_specifier_list '}'

source_refresh_actions
    = 'REFRESH_ACTIONS' '{' actions_specifier_list '}'

source_exit_actions
    = 'EXIT_ACTIONS' '{' actions_specifier_list '}'

```

A.6.26 VALUE_ARRAY

```

array
    = 'ARRAY' Identifier '{' array_attribute_list '}'

array_attribute_list
    = array_attribute
    = array_attribute_list array_attribute

array_attribute
    = array_type                /* M */
    = array_size                 /* M */
    = help                      /* O */
    = label                     /* M */

```



```

= response_codes                                /* O */

array_type
= 'TYPE' Identifier ';'

array_size
= 'NUMBER_OF_ELEMENTS' Integer ';'

A.6.27 VARIABLE

variable
= 'VARIABLE' Identifier '{' variable_attribute_list '}'

variable_attribute_list
= variable_attribute
= variable_attribute_list variable_attribute

variable_attribute
= constant_unit                                /* O */
= default_value                                /* O */
= handling                                     /* O */
= help                                         /* O */
= initial_value                                /* O */
= label                                       /* M */
= post_edit_actions                          /* O */
= post_read_actions                          /* O */
= post_write_actions                         /* O */
= pre_edit_actions                           /* O */
= pre_read_actions                           /* O */
= pre_write_actions                          /* O */
= refresh_actions                           /* O */
= read_timeout                               /* O */
= response_codes                             /* O */
= type                                       /* M */
= validity                                   /* O */
= variable_class                             /* M */
= variable_style                             /* O */
= write_timeout                             /* 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'
= 'CONTAINED'
= 'CORRECTION'
= 'DEVICE'
= 'DIAGNOSTIC'
= 'DIGITAL_INPUT'
= 'DIGITAL_OUTPUT'
= 'DISCRETE_INPUT'
= 'DISCRETE_OUTPUT'
= 'DYNAMIC'
= 'FREQUENCY_INPUT'
= 'FREQUENCY_OUTPUT'
= 'HART'
= 'INPUT'

```

```

    = 'LOCAL'
    = 'LOCAL_DISPLAY'
    = 'OPERATE'
    = 'OUTPUT'
    = 'SERVICE'
    = 'TUNE'

variable_style
    = 'STYLE' string ';'

constant_unit
    = 'CONSTANT_UNIT' string_specifier

default_value
    = 'DEFAULT_VALUE' expr_specifier

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_definition_

handling_definition_
    = handling_keyword
    = handling_definition_ '&' handling_keyword

handling_keyword
    = 'READ'
    = 'WRITE'

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_list
    = '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_list
    = method_specification
    = method_list ',' method_specification

method_specification
    = method_reference
    = method_definition

read_timeout
    = 'READ_TIMEOUT' expr_specifier

write_timeout
    = 'WRITE_TIMEOUT' expr_specifier

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

type
    = 'TYPE' type_specifier

type_specifier
    = arithmetic_type
    = enumerated_type
    = index_type
    = string_type
    = date_time_type

```

```

    = object_reference

arithmetic_type
    = 'INTEGER' arithmetic_option_size arithmetic_options
    = 'UNSIGNED_INTEGER' arithmetic_option_size arithmetic_options
    = 'DOUBLE' arithmetic_options
    = 'FLOAT' arithmetic_options

arithmetic_option_size
    =
    = '(' Integer ')'

arithmetic_options
    = ';'
    = '{' arithmetic_option_list '}'

arithmetic_option_list
    = arithmetic_option
    = arithmetic_option_list arithmetic_option

arithmetic_option
    = arithmetic_default_value
    = arithmetic_initial_value
    = display_format
    = edit_format
    = enumerator_list
    = maximum_value
    = maximum_value_n
    = minimum_value
    = minimum_value_n
    = scaling_factor

arithmetic_default_value
    = 'DEFAULT_VALUE' expr_specifier

arithmetic_initial_value
    = 'INITIAL_VALUE' expr_specifier

display_format
    = 'DISPLAY_FORMAT' string_specifier

edit_format
    = 'EDIT_FORMAT' string_specifier

scaling_factor
    = 'SCALING_FACTOR' expr_specifier

maximum_value
    = 'MAX_VALUE' expr_specifier

maximum_value_n
    = MAX_VALUE_n expr_specifier

minimum_value
    = 'MIN_VALUE' expr_specifier

minimum_value_n
    = MIN_VALUE_n expr_specifier

enumerated_type
    = 'ENUMERATED' enumerated_option_size '{' enumerated_option_list
    '}'
    = 'BIT_ENUMERATED' enumerated_option_size '{'

```

```

    bit_enumerated_option_list '}'

enumerated_option_size
=
= '(' Integer ')'

enumerated_option_list
= enumerated_option
= enumerated_option_list enumerated_option

enumerated_option
= arithmetic_default_value
= arithmetic_initial_value
= enumerators_specifier

enumerators_specifier_list
= enumerators_specifier
= enumerators_specifier_list enumerators_specifier

enumerators_specifier
= 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

enumerator_list
= enumerator
= enumerator_list ',' enumerator

enumerator
= '{' enumerator_value ',' description_string '}'
= '{' enumerator_value ',' description_string ',' help_string '}'

enumerator_value
= Integer
= Real
= string

description_string
= string

help_string
= string

bit_enumerated_option_list
= bit_enumerated_option
= bit_enumerated_option_list bit_enumerated_option

bit_enumerated_option
= arithmetic_default_value
= arithmetic_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 ',' description_string
      '}'
    = '{' Integer ',' description_string ','
      help_string
      '}'
    = '{' Integer ',' description_string ','
      variable_class_definition_wrap
      '}'
    = '{' Integer ',' description_string ','
      status_class
      '}'
    = '{' Integer ',' description_string ','
      help_string ',' variable_class_definition_wrap
      '}'
    = '{' Integer ',' description_string ','
      help_string ',' status_class
      '}'
    = '{' Integer ',' description_string ','
      help_string ',' method_reference
      '}'
    = '{' Integer ',' description_string ','
      variable_class_definition_wrap ',' status_class
      '}'
    = '{' Integer ',' description_string ','
      variable_class_definition_wrap ',' method_reference
      '}'
    = '{' Integer ',' description_string ','
      status_class ',' method_reference
      '}'
    = '{' Integer ',' description_string ','
      help_string ',' variable_class_definition_wrap ',' status_class
      '}'
    = '{' Integer ',' description_string ','
      help_string ',' variable_class_definition_wrap ',' method_reference
      '}'
    = '{' Integer ',' description_string ','
      help_string ',' status_class ',' method_reference
      '}'
    = '{' Integer ',' description_string ','
      variable_class_definition_wrap ',' status_class ',' method_reference

```

```

    '}'
    = '{' Integer ',' description_string ','
        help_string ',' variable_class_definition_wrap ',' status_class ','
method_reference
    '}'

variable_class_definition_wrap
    = variable_class_definition

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'
    = 'IGNORE_IN_TEMPORARY_MASTER'
    = 'BAD'

index_type
    = 'INDEX' item_array_reference ';'
    = 'INDEX' '(' Integer ')' item_array_reference ';'

string_type
    = string_option_type string_option_size ';'
    = string_option_type string_option_size '{' '}'
    = string_option_type string_option_size '{' string_option_list '}'

string_option_type
    = string_keyword

string_keyword
    = 'ASCII'
    = 'BITSTRING'
    = 'EUC'
    = 'PACKED_ASCII'
    = 'PASSWORD'
    = 'VISIBLE'
    = 'OCTET'

string_option_size
    = '(' Integer ')'

string_option_list
    = string_option
    = string_option_list string_option

```

```

string_option
    = string_default_value
    = string_initial_value
    = enumerator_list

string_default_value
    = 'DEFAULT_VALUE' string ';'

string_initial_value
    = 'INITIAL_VALUE' string ';'

date_time_type
    = 'DATE' ';'
    = 'DATE' '{' '}'
    = 'DATE' '{' string_option_list '}'
    = 'DATE_AND_TIME' ';'
    = 'DATE_AND_TIME' '{' '}'
    = 'DATE_AND_TIME' '{' string_option_list '}'
    = 'DURATION' ';'
    = 'DURATION' '{' '}'
    = 'DURATION' '{' string_option_list '}'
    = 'TIME' ';'
    = 'TIME' '{' '}'
    = 'TIME' '{' string_option_list '}'
    = 'TIME' '(' Integer ')' ';'
    = 'TIME' '(' Integer ')' '{' '}'
    = 'TIME' '(' Integer ')' '{' string_option_list '}'
    = 'TIME_VALUE' ';'
    = 'TIME_VALUE' '{' '}'
    = 'TIME_VALUE' '{' string_option_list '}'

object_reference
    = 'OBJECT_REFERENCE' object_reference_value ';'

object_reference_value
    = 'ROOT'
    = 'PARENT'
    = 'CHILD'
    = 'SELF'
    = 'NEXT'
    = 'PREV'
    = 'FIRST'
    = 'LAST'

```

A.6.28 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
    = help                /* O */
    = label                /* O */
    = members              /* M */
    = response_codes       /* O */

```


A.6.29 WAVEFORM

```

waveform
    = 'WAVEFORM' Identifier '{' waveform_attribute_list '}'

waveform_attribute_list
    = waveform_attribute_listR

waveform_attribute_listR
    = waveform_attribute
    = waveform_attribute_listR waveform_attribute

waveform_attribute
    = waveform_help                /* O */
    = waveform_label               /* O */
    = waveform_type                /* M */
    = waveform_handling            /* O */
    = waveform_emphasis            /* O */
    = waveform_exit_actions        /* O */
    = waveform_line_color          /* O */
    = waveform_line_type           /* O */
    = waveform_y_axis              /* O */
    = waveform_init_actions        /* O */
    = waveform_refresh_actions     /* O */
    = waveform_key_points          /* O */

waveform_help
    = help

waveform_label
    = label

waveform_handling
    = handling

waveform_emphasis
    = 'EMPHASIS' boolean_specifier

waveform_exit_actions
    = 'EXIT_ACTIONS' '{' actions_specifier_list '}'

waveform_line_color
    = 'LINE_COLOR' line_color_specifier

waveform_line_type
    = 'LINE_TYPE' line_type_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_listR

line_color_slection_listR
    =
    = line_color_selection_listR line_color_selection line_color_selection

```

```

line_color_selection
    = 'CASE' expr ':' line_color_specifier
    = 'DEFAULT' ':' line_color_specifier

color
    = Integer
    = variable_reference

line_type_specifier
    = line_type_keyword ';'
    = 'IF' '(' expr ')' '{' line_type_specifier '}'
    = 'IF' '(' expr ')' '{' stmt1: line_type_specifier '}'
    = 'ELSE' '{' stmt2: line_type_specifier '}'
    = 'SELECT' '(' expr ')' '{' line_type_selection_list '}'

line_type_selection_list
    = line_type_selection_listR

line_type_selection_listR
    = line_type_selection
    = line_type_selection_listR line_type_selection

line_type_selection
    = 'CASE' expr ':' line_type_specifier
    = 'DEFAULT' ':' line_type_specifier

line_type_keyword
    = 'DATA'
    = 'DATA' digit
    = 'LOW_LOW_LIMIT'
    = 'LOW_LIMIT'
    = 'HIGH_LIMIT'
    = 'HIGH_HIGH_LIMIT'
    = 'TRANSPARENT'

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_listR

waveform_yt_type_option_listR
    = waveform_yt_type_option
    = waveform_yt_type_option_listR 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_listR

values_option_listR
    = values_option
    = values_option_listR ',' 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_listR

waveform_xy_type_option_listR
    = waveform_xy_type_option
    = waveform_xy_type_option_listR 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 ')' '{' stmt1: axis_specifier '}'
    'ELSE' '{' stmt2: axis_specifier '}'
    = 'SELECT' '(' expr ')' '{' axis_selection_list '}'

axis_selection_list
    = axis_selection_listR

axis_selection_listR
    = axis_selection
    = axis_selection_listR axis_selection

axis_selection

```

```

    = 'CASE' expr ':' axis_specifier
    = 'DEFAULT' ':' axis_specifier

waveform_init_actions
    = 'INIT_ACTIONS' '{' actions_specifier_list '}'

waveform_refresh_actions
    = 'REFRESH_ACTIONS' '{' actions_specifier_list '}'

waveform_key_points
    = 'KEY_POINTS' '{' key_points_options_list '}'

key_points_options_list
    = key_points_options_listR

key_points_options_listR
    = key_points_options
    = key_points_options_listR key_points_options

key_points_options
    = waveform_x_values          /* M */
    = waveform_y_values          /* M */

```

A.6.30 Attributs communs

```

help
    = 'HELP' string_specifier

label
    = 'LABEL' string_specifier

members
    = 'MEMBERS' '{' members_specifier_list '}'

number_specifier:
    = number_items ';'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    = 'IF' '(' expr ')' '{' number_specifier '}'
    'ELSE' '{' number_specifier '}'
    = 'SELECT' '(' expr ')' '{' number_selection_list '}'

number_items:
    = expr
    = variable_reference

response_codes
    = 'RESPONSE_CODES' '{' response_codes_specifier_list '}'
    = 'RESPONSE_CODES' response_code_reference_specifier

validity
    = 'VALIDITY' boolean_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_literal
    = variable_reference
    = variable_reference '(' Integer ')'
    = dictionary_reference

string_literal
    = String
    = string_literal String

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 ',' description_string ';'
    = Identifier ',' reference_without_call ',' description_string ','

help_string
    ';'

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

response_code_reference_selection
    = 'CASE' expr ':' response_code_reference_specifier
    = 'DEFAULT' ':' response_code_reference_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'

```

A.6.31 OPEN, CLOSE

```

open
    = 'OPEN' filename

```

```

close
    = 'CLOSE' filename

```

```

filename
    = Identifier

```

A.6.32 Expression

```

primary_expr
    = reference
    = conditional_expr
    = String
    = 'PI'
    = '(' expr ')'
    = primary_expr_keyword

```

```

primary_expr_keyword
    = axis_scaling_keyword

```

```

postfix_expr
    = primary_expr
    = postfix_expr '++'
    = postfix_expr '--'

```

```

unary_expr
    = postfix_expr
    = '++' unary_expr
    = '--' unary_expr
    = unary_operator unary_expr
    = Identifier '(' ')'
    = Identifier '(' c_argument_expr_list ')'
    = 'CURRENT_ROLE'

```

```

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.6.33 C-Grammer

```

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
= ExprInteger

```

```

c_real
= ExprReal

```

```

c_string
= c_string_literal

```

```

c_string_literal
= string_literal

```

```

c_reference
= reference

```

```

c_postfix_expr
= c_primary_expr
= c_postfix_expr '[' c_expr ']'
= Identifier '(' ')'
= Identifier '(' c_argument_expr_list ')'
= c_postfix_expr '.' Identifier
= c_postfix_expr '.' selector
= c_postfix_expr '++'
= c_postfix_expr '--'

```

```

c_argument_expr_list
= c_assignment_expr
= c_argument_expr_list ',' c_assignment_expr

```

```

c_unary_expr
= c_postfix_expr
= '++' c_unary_expr
= '--' c_unary_expr
= c_unary_operator c_postfix_expr

```



```

c_unary_operator
    = '+'
    = '-'
    = '~'
    = '!'

c_multiplicative_expr
    = c_unary_expr
    = c_multiplicative_expr '*' c_unary_expr
    = c_multiplicative_expr '/' c_unary_expr
    = c_multiplicative_expr '%' c_unary_expr

c_additive_expr
    = c_multiplicative_expr
    = c_additive_expr '+' c_multiplicative_expr
    = c_additive_expr '-' c_multiplicative_expr

c_shift_expr
    = c_additive_expr
    = c_shift_expr '<<' c_additive_expr
    = c_shift_expr '>>' c_additive_expr

c_relational_expr
    = c_shift_expr
    = c_relational_expr '<' c_shift_expr
    = c_relational_expr '>' c_shift_expr
    = c_relational_expr '<=' c_shift_expr
    = c_relational_expr '>=' c_shift_expr

c_equality_expr
    = c_relational_expr
    = c_equality_expr '==' c_relational_expr
    = c_equality_expr '!=' c_relational_expr

c_and_expr
    = c_equality_expr
    = c_and_expr '&' c_equality_expr

c_exclusive_or_expr
    = c_and_expr
    = c_exclusive_or_expr '^' c_and_expr

c_inclusive_or_expr
    = c_exclusive_or_expr
    = c_inclusive_or_expr '|' c_exclusive_or_expr

c_logical_and_expr
    = c_inclusive_or_expr
    = c_logical_and_expr '&&' c_inclusive_or_expr

c_logical_or_expr
    = c_logical_and_expr
    = c_logical_or_expr '||' c_logical_and_expr

c_conditional_expr
    = c_logical_or_expr
    = c_logical_or_expr '?' c_logical_or_expr ':' c_conditional_expr

c_assignment_expr
    = c_conditional_expr
    = c_unary_expr c_assignment_operator c_assignment_expr

c_assignment_operator

```

```

    = '='
    = '*='
    = '/='
    = '%='
    = '+='
    = '-='
    = '>>='
    = '<<='
    = '&='
    = '^='
    = '|='

c_expr
    = c_assignment_expr
    = c_expr ',' c_assignment_expr

c_constant_expr
    = c_conditional_expr

c_declaration
    = c_declaration_specifiers c_declarator_list ';'

c_declaration_specifiers
    = c_type_specifier
    = c_declaration_specifiers c_type_specifier

c_type_specifier
    = 'char'
    = 'short'
    = 'int'
    = 'long'
    = 'signed'
    = 'unsigned'
    = 'float'
    = 'double'
    = 'DD_STRING'

c_declarator_list
    = c_declarator
    = c_declarator_list ',' c_declarator

c_declarator
    = Identifier
    = Identifier c_declarator_array_specifier_list

c_declarator_array_specifier_list
    = c_declarator_array_specifier
    = c_declarator_array_specifier_list c_declarator_array_specifier

c_declarator_array_specifier
    = '[' ']'
    = '[' c_constant_expr ']'

c_statement
    = c_labeled_statement
    = c_compound_statement
    = c_expr_statement
    = c_selection_statement
    = c_iteration_statement
    = c_jump_statement

c_labeled_statement
    = 'case' c_constant_expr ':' c_statement

```

```

    = 'default' ':' c_statement

c_compound_statement
    = '{' '}'
    = '{' c_statement_list '}'
    = '{' c_declaration_list '}'
    = '{' c_declaration_list c_statement_list '}'

c_declaration_list
    = c_declaration
    = c_declaration_list c_declaration

c_statement_list
    = c_statement
    = c_statement_list c_statement

c_expr_statement
    = ';'
    = c_expr ';'

c_selection_statement
    = 'if' '(' c_expr ')' c_statement
    = 'if' '(' c_expr ')' c_statement 'else' c_statement
    = 'switch' '(' c_expr ')' c_statement

c_iteration_statement
    = 'do' c_statement 'while' '(' c_expr ')' ';'
    = 'while' '(' c_expr ')' c_statement ';'
    = 'for' '(' ';' ';' ';' ')'
        c_statement
    = 'for' '(' ';' ';' c_expr ')'
        c_statement
    = 'for' '(' ';' c_expr ';' ')'
        c_statement
    = 'for' '(' ';' c_expr ';' c_expr ')'
        c_statement
    = 'for' '(' c_expr ';' ';' ')'
        c_statement
    = 'for' '(' c_expr ';' ';' c_expr ')'
        c_statement
    = 'for' '(' c_expr ';' c_expr ';' ')'
        c_statement
    = 'for' '(' c_expr ';' c_expr ';' c_expr ')'
        c_statement

c_jump_statement
    = 'continue' ';'
    = 'break' ';'
    = 'return' ';'
    = 'return' c_expr ';'

```

A.6.34 Redéfinition

```

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_type_redefinition
    = array_size_redefinition
    = help_redefinition
    = required_label_redefinition
    = response_codes_reference_redefinition

array_type_redefinition
    = 'REDEFINE' array_type

array_size_redefinition
    = 'REDEFINE' array_size

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
    = optional_label_redefinition
    = minimum_value_redefinition
    = maximum_value_redefinition
    = axis_scaling_redefinition
    = help_redefinition
    = constant_unit_redefinition

axis_scaling_redefinition
    = 'DELETE' 'SCALING' ';'
    = 'REDEFINE' axis_scaling

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_axis_items_redefinition
    = block_a_characteristics_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
    = block_a_help_redefinition
    = block_a_image_items_redefinition
    = block_a_item_array_items_redefinition
    = block_a_label_redefinition
    = block_a_local_parameters_redefinition
    = block_a_list_items_redefinition
    = block_a_menu_items_redefinition
    = block_a_method_items_redefinition
    = block_a_parameters_redefinition
    = block_a_parameter_lists_redefinition

```

```

= block_a_refresh_items_redefinition
= block_a_source_items_redefinition
= block_a_unit_items_redefinition
= block_a_wao_items_redefinition
= block_a_waveform_items_redefinition
= block_b_number_redefinition
= block_b_type_redefinition

block_a_axis_items_redefinition
= 'DELETE' 'AXIS_ITEMS' ';'
= 'REDEFINE' block_a_axis_items

block_a_characteristics_redefinition
= 'REDEFINE' block_a_characteristics

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_help_redefinition
= help_redefinition

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_label_redefinition
= required_label_redefinition

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_parameters_redefinition
    = 'PARAMETERS' '{' parameter_redefinition_list '}'

parameter_redefinition_list
    = parameter_redefinition
    = parameter_redefinition_list parameter_redefinition

parameter_redefinition
    = 'REDEFINE' member
    = 'ADD' member

block_a_parameter_lists_redefinition
    = 'PARAMETER_LISTS' '{' parameter_redefinition_list '}'

block_a_refresh_items_redefinition
    = 'DELETE' 'REFRESH_ITEMS' ';'
    = 'REDEFINE' block_a_refresh_items

block_a_source_items_redefinition
    = 'DELETE' 'SOURCE_ITEMS' ';'
    = 'REDEFINE' block_a_source_items

block_a_unit_items_redefinition
    = 'DELETE' 'UNIT_ITEMS' ';'
    = 'REDEFINE' block_a_unit_items

block_a_wao_items_redefinition
    = 'DELETE' 'WRITE_AS_ONE_ITEMS' ';'
    = 'REDEFINE' block_a_wao_items

block_a_waveform_items_redefinition
    = 'DELETE' 'WAVEFORM_ITEMS' ';'
    = 'REDEFINE' block_a_waveform_items

block_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
    = optional_label_redefinition
    = help_redefinition
    = members_redefinition
    = chart_height_redefinition
    = chart_width_redefinition
    = chart_type_redefinition
    = chart_cycle_time_redefinition
    = chart_length_redefinition

```

```

chart_height_redefinition
    = 'DELETE' 'HEIGHT' ';'
    = 'REDEFINE' chart_height

chart_width_redefinition
    = 'DELETE' 'WIDTH' ';'
    = 'REDEFINE' chart_width

chart_type_redefinition
    = 'DELETE' 'TYPE' ';'
    = 'REDEFINE' chart_type

chart_cycle_time_redefinition
    = 'DELETE' 'CYCLE_TIME' ';'
    = 'REDEFINE' chart_cycle_time

chart_length_redefinition
    = 'DELETE' 'LENGTH' ';'
    = 'REDEFINE' chart_length

collection_redefinition
    = 'DELETE' 'COLLECTION' Identifier ';'
    = 'REDEFINE' 'COLLECTION' Identifier '{' '}' = 'REDEFINE'
'COLLECTION' 'OF' item_type Identifier '{' '}'
    = 'REDEFINE' 'COLLECTION' Identifier '{' collection_attribute_list '}'
    = 'REDEFINE' 'COLLECTION' 'OF' item_type Identifier '{'
collection_attribute_list '}'
    = 'COLLECTION' Identifier '{' collection_attribute_redefinition_list
'}'
    = 'COLLECTION' 'OF' item_type Identifier '{'
collection_attribute_redefinition_list '}'

collection_attribute_redefinition_list
    = collection_attribute_redefinition
    = collection_attribute_redefinition_list

collection_attribute_redefinition
    = help_redefinition
    = optional_label_redefinition
    = members_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_header_redefinition
    = command_slot_redefinition
    = command_index_redefinition
    = command_block_redefinition
    = command_number_redefinition
    = command_operation_redefinition
    = command_transaction_redefinition
    = command_connection_redefinition
    = command_module_redefinition

```

```

= command_response_codes_redefinition

command_header_redefinition
= 'DELETE' 'HEADER' ';'
= 'REDEFINE' command_header

command_slot_redefinition
= 'DELETE' 'SLOT' ';'
= 'REDEFINE' command_slot

command_index_redefinition
= 'DELETE' 'INDEX' ';'
= 'REDEFINE' command_index

command_block_redefinition
= 'DELETE' 'BLOCK' ';'
= 'REDEFINE' command_block

command_number_redefinition
= 'DELETE' 'NUMBER' ';'
= 'REDEFINE' command_number

command_operation_redefinition
= 'REDEFINE' command_operation

command_transaction_redefinition
= 'ADD' command_transaction
= 'REDEFINE' command_transaction
= 'DELETE' 'TRANSACTION' ';'
= 'DELETE' 'TRANSACTION' Integer '...'

command_connection_redefinition
= 'DELETE' 'CONNECTION' ';'
= 'REDEFINE' command_connection

command_module_redefinition
= 'DELETE' 'MODULE' ';'
= 'REDEFINE' command_module

command_response_codes_redefinition
= response_codes_reference_redefinition

connection_redefinition
= 'DELETE' 'CONNECTION' Identifier ';'
= 'REDEFINE' 'CONNECTION' Identifier '{' '}'
= 'REDEFINE' 'CONNECTION' Identifier '{'
  connection_attribute_list '}'
= 'CONNECTION' Identifier '{'
  connection_attribute_redefinition_list '}'

connection_attribute_redefinition_list
= connection_attribute_redefinition
= connection_attribute_redefinition_list

connection_attribute_redefinition
= appinstance_redefinition

appinstance_redefinition
= 'REDEFINE' appinstance

domain_redefinition
= 'DELETE DOMAIN' Identifier ';'

```



```

= 'REDEFINE DOMAIN' Identifier '{' domain_attribute_list '}'
= 'DOMAIN' Identifier '{' domain_attribute_redefinition_list '}'

domain_attribute_redefinition_list
= domain_attribute_redefinition
= domain_attribute_redefinition_list domain_attribute_redefinition

domain_attribute_redefinition
= handling_redefinition
= response_codes_reference_redefinition

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
= display_items_redefinition
= edit_items_redefinition
= required_label_redefinition
= post_edit_actions_redefinition
= pre_edit_actions_redefinition

display_items_redefinition
= 'DELETE' 'DISPLAY_ITEMS' ';'
= 'REDEFINE' display_items

edit_items_redefinition
= 'REDEFINE' edit_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
= optional_label_redefinition
= help_redefinition
= members_redefinition

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
= optional_label_redefinition

```

```
= help_redefinition
= members_redefinition
= graph_x_axis_redefinition
= graph_height_redefinition
= graph_width_redefinition

= graph_cycle_time_redefinition

graph_x_axis_redefinition
= 'DELETE' 'X_AXIS' ';'
= 'REDEFINE' waveform_x_axis

graph_height_redefinition
= 'DELETE' 'HEIGHT' ';'
= 'REDEFINE' graph_height

graph_width_redefinition
= 'DELETE' 'WIDTH' ';'
= 'REDEFINE' graph_width

graph_cycle_time_redefinition
= 'DELETE' 'CYCLE_TIME' ';'
= 'REDEFINE' graph_cycle_time

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
= optional_label_redefinition
= help_redefinition
= grid_vectors_redefinition
= grid_handling_redefinition
= grid_height_redefinition
= grid_width_redefinition
= grid_orientation_redefinition
= grid_validity_redefinition

grid_vectors_redefinition
= 'REDEFINE' grid_vectors

grid_handling_redefinition
= 'DELETE' 'HANDLING' ';'
= 'REDEFINE' grid_handling

grid_height_redefinition
= 'DELETE' 'HEIGHT' ';'
= 'REDEFINE' grid_height

grid_width_redefinition
= 'DELETE' 'WIDTH' ';'
= 'REDEFINE' grid_width

grid_orientation_redefinition
= 'DELETE' 'ORIENTATION' ';'
= 'REDEFINE' grid_orientation

grid_validity_redefinition
```

```

= 'DELETE' 'VALIDITY' ';'
= 'REDEFINE' grid_validity

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
= optional_label_redefinition
= help_redefinition
= image_path_redefinition
= image_link_redefinition
= image_validity_redefinition

image_path_redefinition
= 'REDEFINE' image_path

image_link_redefinition
= 'DELETE' 'LINK' ';'
= 'REDEFINE' image_link

image_validity_redefinition
= 'DELETE' 'VALIDITY' ';'
= 'REDEFINE' image_validity

item_array_redefinition
= 'DELETE' 'ITEM_ARRAY' Identifier ';'
= 'REDEFINE' 'ITEM_ARRAY' 'OF' item_type Identifier '{' '}'
= 'REDEFINE' 'ITEM_ARRAY' 'OF' item_type Identifier '{'
item_array_attribute_list '}'
= 'ITEM_ARRAY' 'OF' 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
= elements_redefinition
= help_redefinition
= optional_label_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
    = help_redefinition
    = optional_label_redefinition
    = list_capacity_redefinition
    = list_count_redefinition
    = list_type_redefinition

list_capacity_redefinition
    = 'DELETE' 'CAPACITY' ';'
    = 'REDEFINE' list_capacity

list_count_redefinition
    = 'DELETE' 'COUNT' ';'
    = 'REDEFINE' list_count

list_type_redefinition
    = 'REDEFINE' list_type

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
    = help_redefinition
    = required_label_redefinition
    = menu_access_redefinition
    = menu_entry_redefinition
    = menu_items_redefinition
    = menu_purpose_redefinition
    = menu_role_redefinition
    = menu_style_redefinition
    = validity_redefinition
    = pre_edit_actions_redefinition
    = pre_read_actions_redefinition
    = pre_write_actions_redefinition
    = post_edit_actions_redefinition
    = post_read_actions_redefinition
    = post_write_actions_redefinition

menu_access_redefinition
    = 'DELETE' 'ACCESS' ';'
    = 'REDEFINE' menu_access

menu_entry_redefinition
    = 'DELETE' 'ENTRY' ';'
    = 'REDEFINE' menu_entry

menu_items_redefinition
    = 'REDEFINE' menu_items

menu_purpose_redefinition

```

```

= 'DELETE' 'PURPOSE' ';'
= 'REDEFINE' menu_purpose

menu_role_redefinition
= 'DELETE' 'ROLE' ';'
= 'REDEFINE' menu_role

menu_style_redefinition
= 'DELETE' 'STYLE' ';'
= 'REDEFINE' menu_style

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
= help_redefinition
= required_label_redefinition
= method_access_redefinition
= method_definition_redefinition
= validity_redefinition
= variable_class_redefinition

method_access_redefinition
= 'REDEFINE' method_access

method_definition_redefinition
= 'REDEFINE' method_definition

program_redefinition
= 'DELETE' 'PROGRAM' Identifier ';'
= 'REDEFINE' 'PROGRAM' Identifier '{' '}'
= 'REDEFINE' 'PROGRAM' Identifier '{' program_attribute_list '}'
= 'PROGRAM' Identifier '{' program_attribute_redefinition_list
  '}'

program_attribute_redefinition_list
= program_attribute_redefinition
= program_attribute_redefinition_list program_attribute_redefinition

program_attribute_redefinition
= arguments_redefinition
= response_codes_reference_redefinition

arguments_redefinition
= 'DELETE' 'ARGUMENTS' ';'
= 'REDEFINE' arguments

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
    = help_redefinition
    = required_label_redefinition
    = members_redefinition
    = response_codes_reference_redefinition

refresh_relation_redefinition
    = 'DELETE' 'REFRESH' Identifier ';'
    = 'REDEFINE' 'REFRESH' Identifier
      '{' '}'
    = 'REDEFINE' 'REFRESH' Identifier
      '{' refresh_relation_left ':' refresh_relation_right '}'

unit_relation_redefinition
    = 'DELETE' 'UNIT' Identifier ';'
    = 'REDEFINE' 'UNIT' Identifier
      '{' '}'
    = 'REDEFINE' 'UNIT' Identifier
      '{' unit_relation_left ':' unit_relation_right '}'

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

help_redefinition
    = 'DELETE' 'HELP' ';'
    = 'REDEFINE' help

required_label_redefinition
    = 'REDEFINE' label

optional_label_redefinition
    = 'DELETE' 'LABEL' ';'
    = 'REDEFINE' label

response_codes_reference_redefinition
    = 'DELETE' 'RESPONSE_CODES' ';'
    = 'REDEFINE' response_codes

validity_redefinition
    = 'DELETE' 'VALIDITY' ';'
    = 'REDEFINE' validity

```

```

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
    = help_redefinition
    = optional_label_redefinition
    = source_emphasis_redefinition
    = source_line_color_redefinition
    = source_line_type_redefinition
    = source_y_axis_redefinition
    = members_redefinition
    = source_refresh_actions_redefinition
    = source_init_actions_redefinition
    = source_exit_actions_redefinition

source_emphasis_redefinition
    = 'DELETE' 'EMPHASIS' ';'
    = 'REDEFINE' source_emphasis

source_line_color_redefinition
    = 'DELETE' 'LINE_COLOR' ';'
    = 'REDEFINE' source_line_color

source_line_type_redefinition
    = 'DELETE' 'LINE_TYPE' ';'
    = 'REDEFINE' source_line_type

source_y_axis_redefinition
    = 'DELETE' 'Y_AXIS' ';'
    = 'REDEFINE' waveform_y_axis

source_init_actions_redefinition
    = 'DELETE' 'INIT_ACTIONS' ';'
    = 'REDEFINE' source_init_actions

source_refresh_actions_redefinition
    = 'DELETE' 'REFRESH_ACTIONS' ';'
    = 'REDEFINE' source_refresh_actions

source_exit_actions_redefinition
    = 'DELETE' 'REFRESH_ACTIONS' ';'
    = 'REDEFINE' source_refresh_actions

variable_redefinition

```

```

= 'DELETE' 'VARIABLE' Identifier ';'
= 'REDEFINE' 'VARIABLE' Identifier '{' variable_attribute_list
  '}'
= 'VARIABLE' Identifier '{' variable_attribute_redefinition_list
  '}'

variable_attribute_redefinition_list
=
= variable_attribute_redefinition_list variable_attribute_redefinition

variable_attribute_redefinition
= constant_unit_redefinition
= default_value_redefinition
= handling_redefinition
= help_redefinition
= initial_value_redefinition
= required_label_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
= refresh_actions_redefinition
= read_timeout_redefinition
= response_codes_reference_redefinition
= type_redefinition
= validity_redefinition
= variable_class_redefinition
= variable_style_redefinition
= write_timeout_redefinition

variable_class_redefinition
= 'REDEFINE' variable_class

variable_style_redefinition
= 'DELETE' 'STYLE' ';'
= 'REDEFINE' variable_style

constant_unit_redefinition
= 'DELETE' 'CONSTANT_UNIT' ';'
= 'REDEFINE' constant_unit

default_value_redefinition
= 'DELETE' 'DEFAULT_VALUE' ';'
= 'REDEFINE' default_value

handling_redefinition
= 'DELETE' 'HANDLING' ';'
= 'REDEFINE' handling

initial_value_redefinition
= 'DELETE' 'INITIAL_VALUE' ';'
= 'REDEFINE' initial_value

pre_edit_actions_redefinition
= 'DELETE' 'PRE_EDIT_ACTIONS' ';'
= 'REDEFINE' pre_edit_actions

post_edit_actions_redefinition
= 'DELETE' 'POST_EDIT_ACTIONS' ';'
= 'REDEFINE' post_edit_actions

```



```

pre_read_actions_redefinition
    = 'DELETE' 'PRE_READ_ACTIONS' ';'
    = 'REDEFINE' pre_read_actions

post_read_actions_redefinition
    = 'DELETE' 'POST_READ_ACTIONS' ';'
    = 'REDEFINE' post_read_actions

pre_write_actions_redefinition
    = 'DELETE' 'PRE_WRITE_ACTIONS' ';'
    = 'REDEFINE' pre_write_actions

post_write_actions_redefinition
    = 'DELETE' 'POST_WRITE_ACTIONS' ';'
    = 'REDEFINE' post_write_actions

refresh_actions_redefinition
    = 'DELETE' 'REFRESH_ACTIONS' ';'
    = 'REDEFINE' refresh_actions

read_timeout_redefinition
    = 'DELETE' 'READ_TIMEOUT' ';'
    = 'REDEFINE' read_timeout

write_timeout_redefinition
    = 'DELETE' 'WRITE_TIMEOUT' ';'
    = 'REDEFINE' write_timeout

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

arithmetic_type_redefinition
    = 'INTEGER' arithmetic_option_size_redefinition '{'
        arithmetic_option_redefinition_list '}'
    = 'UNSIGNED_INTEGER' arithmetic_option_size_redefinition '{'
        arithmetic_option_redefinition_list '}'
    = 'FLOAT' 'N' arithmetic_option_redefinition_list '}'
    = 'DOUBLE' '{' arithmetic_option_redefinition_list '}'

arithmetic_option_size_redefinition
    = arithmetic_option_size_redefinition1

arithmetic_option_size_redefinition1
    =
    = '(' Integer ')'

arithmetic_option_redefinition_list
    = arithmetic_option_redefinition
    = arithmetic_option_redefinition_list arithmetic_option_redefinition

arithmetic_option_redefinition
    = arithmetic_default_value_redefinition
    = arithmetic_initial_value_redefinition
    = display_format_redefinition
    = edit_format_redefinition
    = enumerator_list_redefinition
    = maximum_value_redefinition
    = maximum_value_n_redefinition

```

```

    = minimum_value_redefinition
    = minimum_value_n_redefinition
    = scaling_factor_redefinition

arithmetic_default_value_redefinition
    = 'DELETE' 'DEFAULT_VALUE' ';'
    = 'REDEFINE' arithmetic_default_value

arithmetic_initial_value_redefinition
    = 'DELETE' 'INITIAL_VALUE' ';'
    = 'REDEFINE' arithmetic_initial_value

display_format_redefinition
    = 'DELETE' 'DISPLAY_FORMAT' ';'
    = 'REDEFINE' display_format

edit_format_redefinition
    = 'DELETE' 'EDIT_FORMAT' ';'
    = 'REDEFINE' edit_format

enumerator_list_redefinition
    = 'DELETE' Integer ';'
    = 'REDEFINE' enumerator
    = 'ADD' enumerator

maximum_value_redefinition
    = 'DELETE' 'MAX_VALUE' ';'
    = 'REDEFINE' maximum_value

maximum_value_n_redefinition
    = 'DELETE' MAX_VALUE_n ';'
    = 'REDEFINE' maximum_value_n

minimum_value_redefinition
    = 'DELETE' 'MIN_VALUE' ';'
    = 'REDEFINE' minimum_value

minimum_value_n_redefinition
    = 'DELETE' MIN_VALUE_n ';'
    = 'REDEFINE' minimum_value_n

scaling_factor_redefinition
    = 'DELETE' 'SCALING_FACTOR' ';'
    = 'REDEFINE' scaling_factor

enumerated_type_redefinition
    = 'ENUMERATED' enumerated_option_size_redefinition '{'
        enumerated_option_redefinition_list '}'
    = 'BIT_ENUMERATED' enumerated_option_size_redefinition '{'
        bit_enumerated_option_redefinition_list '}'

enumerated_option_size_redefinition
    = enumerated_option_size_redefinition1

enumerated_option_size_redefinition1
    =
    = '(' Integer ')'

enumerated_option_redefinition_list
    = enumerated_option_redefinition
    = enumerated_option_redefinition_list enumerated_option_redefinition

enumerated_option_redefinition

```

```

= arithmetic_default_value_redefinition
= arithmetic_initial_value_redefinition
= enumerators_redefinition

enumerators_redefinition
= 'DELETE' Integer ';'
= 'REDEFINE' enumerator
= 'ADD' enumerator

bit_enumerated_option_redefinition_list
= bit_enumerated_option_redefinition
= bit_enumerated_option_redefinition_list
bit_enumerated_option_redefinition

bit_enumerated_option_redefinition
= arithmetic_default_value_redefinition
= arithmetic_initial_value_redefinition
= bit_enumerators_redefinition

bit_enumerators_redefinition
= 'DELETE' Integer ';'
= 'REDEFINE' bit_enumerator
= 'ADD' bit_enumerator

string_type_redefinition
= string_option_type '{' string_option_redefinition_list '}'
= string_option_type string_option_size_redefinition '{'
string_option_redefinition_list '}'

string_option_size_redefinition
= string_option_size_redefinition1

string_option_size_redefinition1
= '(' Integer ')'

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_list_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_list_redefinition
= 'DELETE' string ';'
= 'REDEFINE' enumerator
= 'ADD' enumerator

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
= help_redefinition
= optional_label_redefinition
= members_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
= help_redefinition
= optional_label_redefinition
= waveform_type_redefinition
= handling_redefinition
= waveform_emphasis_redefinition
= waveform_exit_actions_redefinition
= waveform_line_color_redefinition
= waveform_line_type_redefinition
= waveform_y_axis_redefinition
= waveform_init_actions_redefinition
= waveform_refresh_actions_redefinition
= waveform_key_points_redefinition

waveform_type_redefinition
= 'REDEFINE' waveform_type
= 'TYPE' waveform_type_redefinition

waveform_type_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
= 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_emphasis_redefinition
    = 'DELETE' 'EMPHASIS' ';'
    = 'REDEFINE' waveform_emphasis

waveform_exit_actions_redefinition
    = 'DELETE' 'EXIT_ACTIONS' ';'
    = 'REDEFINE' waveform_exit_actions

waveform_line_color_redefinition
    = 'DELETE' 'LINE_COLOR' ';'
    = 'REDEFINE' waveform_line_color

waveform_line_type_redefinition
    = 'DELETE' 'LINE_TYPE' ';'
    = 'REDEFINE' waveform_line_type

waveform_y_axis_redefinition
    = 'DELETE' 'Y_AXIS' ';'
    = 'REDEFINE' waveform_y_axis

waveform_init_actions_redefinition
    = 'DELETE' 'INIT_ACTIONS' ';'
    = 'REDEFINE' waveform_init_actions

waveform_refresh_actions_redefinition
    = 'DELETE' 'REFRESH_ACTIONS' ';'
    = 'REDEFINE' waveform_refresh_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_listR

key_points_options_redefinition_listR
    = key_points_options_redefinition
    = key_points_options_redefinition_listR key_points_options_redefinition

key_points_options_redefinition
    = waveform_x_values_redefinition

```

```
= waveform_y_values_redefinition
```

A.6.35 Références

```
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
```

```
selector
```

```
= 'CAPACITY'
= 'CONSTANT_UNIT'
= 'COUNT'
= 'DEFAULT_VALUE'
= 'FIRST'
= 'HELP'
= 'INITIAL_VALUE'
= 'LABEL'
= 'LAST'
= 'MAX_VALUE'
= MAX_VALUE_n
= 'MIN_VALUE'
= MIN_VALUE_n
= 'SCALING'
= 'SCALING_FACTOR'
= 'VARIABLE_STATUS'
= 'VIEW_MAX'
= 'VIEW_MIN'
= 'X_AXIS'
= 'Y_AXIS'
```

```
dictionary_reference
```

```
= '[' Identifier ']'
```

```
axis_reference
```

```
= reference_without_call
```

```
block_reference
```

```
= reference_without_call
```

```
chart_reference
```

```
= reference_without_call
```

```
collection_reference
```

```
= reference_without_call
```

```
connection_reference
```

```
= reference_without_call
```

```
edit_display_reference
```

```
= reference_without_call
```

```
file_reference
```

```
= reference_without_call
```

```
graph_reference
```

```
= reference_without_call
```

```
list_reference
```

```
= reference_without_call
```

```
item_array_reference
```

```
= reference_without_call
```

```
menu_reference
```

```
= reference_without_call
```

```
method_reference
```

```
= reference_without_call
```

```
record_reference
    = reference_without_call
refresh_reference
    = reference_without_call
response_code_reference
    = reference_without_call
source_reference
    = reference_without_call
unit_reference
    = reference_without_call
variable_reference
    = reference_without_call
wao_reference
    = reference_without_call
waveform_reference
    = reference_without_call
```

IECNORM.COM : Click to view the full PDF of IEC 61804-3:2006

Annexe B (normative)

Bibliothèque EDDL Builtin

B.1 Généralités

La présente annexe décrit les sous-routines Builtin, qui peuvent être utilisées dans une EDD. Les Builtins prennent en charge les interactions de l'utilisateur ou d'appareil. Les interactions utilisateur classiques sont, par exemple, des entrées utilisateur de valeurs VARIABLE ou des acquittements. Les interactions classiques d'appareil sont, par exemple, des commandes envoyées à l'appareil ou des codes de réponse d'interprétation et des octets d'état. La présente annexe contient la bibliothèque de Builtins.

Outre les Builtins spécifiés dans le présent document, des Builtins spécifiques à l'utilisateur peuvent être utilisés. Ces Builtins doivent être intégrés dans l'application EDD.

NOTE Les Builtins sont présentés dans l'ordre alphabétique.

B.2 Conventions de description de Builtin

Les conventions suivantes s'appliquent aux définitions de Builtin

- Le Tableau B.1 définit le format des tables d'éléments lexicaux des Builtins.
- Le Tableau B.2 définit le contenu des colonnes de Tableau B.1.

Tableau B.1 – Format des tables d'éléments lexicaux de Builtins

Nom de paramètre	Type	Direction	Usage	Description

Tableau B.2 – Contenu de la table d'éléments lexicaux

Colonne	Texte	Signification
Nom du paramètre	<name>	Nom du paramètre formel
	<return>	Représente la valeur renvoyée du Builtin
Type	Type de données	Type de données abstrait (basique ou dérivé) du paramètre formel
Direction	I	Spécifie un paramètre d'entrée
	O	Spécifie un paramètre de sortie
	E/S	Spécifie un paramètre d'entrée et de sortie
	—	A utiliser si aucune valeur n'est renvoyée (VOID)
Usage	m	Paramètre formel obligatoire
	o	Paramètre formel facultatif
	c	Paramètre formel conditionnel
	—	A utiliser si aucune valeur n'est renvoyée (VOID)
Description	<text>	Texte descriptif pour le paramètre
	—	A utiliser si aucune valeur n'est renvoyée (VOID)

B.3 Builtin abort

Objet

Le Builtin abort affiche un message indiquant que la méthode est interrompue et attend un accusé de réception de l'utilisateur. Une fois créé l'accusé de réception, le système exécute des méthodes d'interruption de la liste de méthodes d'interruption, puis quitte la méthode.

Structure lexicale

abort (VOID)

Les éléments lexicaux du Builtin abort sont présentés au Tableau B.3.

Tableau B.3 – Builtin abort

Nom de paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.4 Builtin abort_on_all_comm_errors

Objet

Builtin abort_on_all_comm_errors définit la méthode d'interruption dès la réception d'une erreur de communication.

Si une méthode est démarrée, elle ne s'interrompt pas sur une erreur de communication tant que abort_on_all_comm_errors n'est pas appelé. Cela évite qu'une méthode ne traite des codes d'erreur de communication en interrompant la méthode si elle reçoit une erreur de communication.

Après cet appel, un appel Builtin qui demande une communication qui renvoie une erreur provoque l'interruption de la méthode. Une erreur de communication nulle n'est pas considérée comme une erreur de communication.

Les codes d'erreur de communication sont définis par les consortiums. Leurs chaînes associées peuvent être définies dans un dictionnaire de texte.

Structure lexicale

abort_on_all_comm_errors(VOID)

L'élément lexical du Builtin abort_on_all_comm_errors est présenté au Tableau B.4.

Tableau B.4 – Builtin abort_on_all_comm_errors

Nom de paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.5 Builtin ABORT_ON_ALL_COMM_STATUS

Objet

Le Builtin ABORT_ON_ALL_COMM_STATUS définit tous les bits dans le masque d'interruption de statut de communication. Cela conduit le système à interrompre la méthode en cours si l'appareil renvoie une valeur de statut de communication. Le statut de communication est défini comme étant le premier octet de données renvoyé dans une transaction, lorsque le bit 7 de cet octet est défini.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, de sorte que la nouvelle valeur du masque est valide uniquement pendant la méthode actuelle. Voir le Builtin ABORT_ON_RESPONSE_CODES pour obtenir une liste des masques disponibles et leurs valeurs par défaut.

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

ABORT_ON_ALL_COMM_STATUS (VOID)

L'élément lexical du Builtin ABORT_ON_ALL_COMM_STATUS est présenté au Tableau B.5.

Tableau B.5 – Builtin ABORT_ON_ALL_COMM_STATUS

Nom du paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.6 Builtin ABORT_ON_ALL_DEVICE_STATUS**Objet**

Le Builtin ABORT_ON_ALL_DEVICE_STATUS définit tous les bits dans le masque d'interruption de statut de l'appareil. Cela conduit le système à interrompre la méthode en cours si l'appareil renvoie une valeur de statut de l'appareil. Le statut de l'appareil est défini comme étant le deuxième octet de données retourné dans une transaction.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, de sorte que la nouvelle valeur du masque est valide uniquement pendant la méthode actuelle. Voir la fonction send_command pour l'implémentation des masques. Voir ABORT_ON_RESPONSE_CODE pour une liste des masques disponibles et leurs valeurs par défaut.

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

ABORT_ON_ALL_DEVICE_STATUS (VOID)

L'élément lexical du Builtin ABORT_ON_ALL_DEVICE_STATUS est présenté au Tableau B.6.

Tableau B.6 – Builtin ABORT_ON_ALL_DEVICE_STATUS

Nom du paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.7 Builtin ABORT_ON_ALL_RESPONSE_CODES**Objet**

Le Builtin ABORT_ON_ALL_RESPONSE_CODES définit tous les bits dans le masque d'interruption du code de réponse. Cela conduit le système à interrompre la méthode en cours si l'appareil renvoie une valeur de code de réponse.

Le code de réponse est défini comme étant le premier octet de données retourné dans une transaction, lorsque le bit 7 de cet octet est 0.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, de sorte que la nouvelle valeur du masque est valide uniquement pendant la méthode actuelle. Voir la fonction de commande d'envoi pour la mise en œuvre des masques. Voir ABORT_ON_RESPONSE_CODE pour une liste des masques disponibles et leurs valeurs par défaut.

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

ABORT_ON_ALL_RESPONSE_CODES (VOID)

L'élément lexical du Builtin ABORT_ON_ALL_RESPONSE_CODES est présenté au Tableau B.7.

Tableau B.7 – Builtin ABORT_ON_ALL_RESPONSE_CODES

Nom du paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.8 Builtin abort_on_all_response_codes**Objet**

Le Builtin abort_on_all_response_codes provoque l'interruption de la méthode en cas de réception d'un code de réponse non-nul après l'appel d'un Builtin, donnant lieu à la formulation d'une demande adressée à un appareil. Ce Builtin évite qu'une méthode ne traite les codes de réponse d'erreur. Si une méthode est démarrée, elle ne s'interrompt pas sur un code de réponse tant que ce Builtin n'est pas appelé. Un code de réponse est une valeur représentant une condition d'erreur spécifique à une application. Une valeur de code de réponse nulle n'est pas considérée comme un code de réponse d'erreur. Chaque code de réponse et type de code de réponse est défini dans l'EDDL.

Structure lexicale

abort_on_all_response_codes (VOID)

L'élément lexical du Builtin abort_on_all_response_codes est présenté au Tableau B.8.

Tableau B.8 – Builtin abort_on_all_response_codes

Nom du paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.9 Builtin abort_on_comm_error**Objet**

Provoque l'interruption de la méthode à la réception d'une erreur de communication particulière.

Le Builtin abort_on_comm_error conduit la méthode à s'interrompre en cas de réception d'une erreur de communication particulière.

Si une méthode est démarrée, elle ne s'interrompt pas sur une erreur de communication particulière tant que le Builtin abort_on_comm_error n'est pas appelé.

NOTE Une erreur de communication nulle n'est pas considérée comme telle.

Ce Builtin permet au rédacteur de la méthode de sélectionner les erreurs de communication qu'il convient de traiter par la méthode, et évite qu'une méthode ne traite un code d'erreur de communication particulier.

Les codes d'erreur de communication sont définis par les consortiums. Leurs chaînes associées peuvent être définies dans un dictionnaire de texte.

Structure lexicale

abort_on_comm_error (error)

Les éléments lexicaux du Builtin abort_on_comm_error sont présentés au Tableau B.9.

Tableau B.9 – Builtin abort_on_comm_error

Nom du paramètre	Type	Direction	Usage	Description
erreur	unsigned long	I	m	Spécifie le code d'erreur
<return>	long	O	m	L'un des codes de retour spécifiés au Tableau B.235

B.10 Builtin ABORT_ON_COMM_ERROR**Objet**

Le Builtin ABORT_ON_COMM_ERROR définit le masque sans erreur de communication, de manière à interrompre la méthode en cas d'erreur de communication lors de l'envoi de la commande.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, de sorte que la nouvelle valeur du masque est valide uniquement pendant la méthode actuelle. Voir la fonction send_command pour l'implémentation des masques. Voir le Builtin ABORT_ON_RESPONSE_CODES pour obtenir une liste des masques disponibles et leurs valeurs par défaut.

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

ABORT_ON_COMM_ERROR(VOID)

L'élément lexical du Builtin ABORT_ON_COMM_ERROR est présenté au Tableau B.10.

Tableau B.10 – Builtin ABORT_ON_COMM_ERROR

Nom du paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.11 Builtin ABORT_ON_COMM_STATUS**Objet**

Le Builtin ABORT_ON_COMM_STATUS définit le(s) bit(s) correct(s) dans le masque d'interruption de statut de communication, de sorte que le comm_status_value spécifié provoque l'interruption de la méthode. Le statut de communication est défini comme étant le premier octet de données renvoyé dans une transaction, lorsque le bit 7 de cet octet est 1.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, la nouvelle valeur du masque étant donc valide uniquement pendant la méthode en cours. Voir la fonction de commande d'envoi pour la mise en œuvre des masques. Voir ABORT_ON_RESPONSE_CODE pour une liste des masques disponibles et leurs valeurs par défaut.

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

ABORT_ON_COMM_STATUS(comm_status)

Les éléments lexicaux du Builtin ABORT_ON_COMM_STATUS sont présentés au Tableau B.11.

Tableau B.11 – Builtin ABORT_ON_COMM_STATUS

Nom du paramètre	Type	Direction	Usage	Description
comm_status	unsigned char	l	m	Spécifie un masque de bit pour le statut de communication
<return>	VOID	—	—	—

B.12 Builtin ABORT_ON_DEVICE_STATUS

Objet

Le Builtin ABORT_ON_DEVICE_STATUS définit le(s) bit(s) correct(s) dans le masque d'interruption de statut de l'appareil, de sorte que la valeur de statut de l'appareil spécifiée provoque l'interruption de la méthode. Le statut de l'appareil est défini comme étant le deuxième octet de données retourné dans une transaction.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, de sorte que la nouvelle valeur du masque est valide uniquement pendant la méthode actuelle. Voir la fonction de commande d'envoi pour la mise en œuvre des masques. Voir ABORT_ON_RESPONSE_CODE pour une liste des masques disponibles et leurs valeurs par défaut.

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

ABORT_ON_DEVICE_STATUS(device_status)

Les éléments lexicaux du Builtin ABORT_ON_DEVICE_STATUS sont présentés au Tableau B.12.

Tableau B.12 – Builtin ABORT_ON_DEVICE_STATUS

Nom du paramètre	Type	Direction	Usage	Description
device_status	unsigned char	l	m	Spécifie un masque de bit pour le statut de l'appareil
<return>	VOID	—	—	—

B.13 Builtin ABORT_ON_NO_DEVICE

Objet

Le Builtin ABORT_ON_NO_DEVICE définit le masque sans appareil, de manière à interrompre la méthode si aucun appareil n'est trouvé lors de l'envoi de la commande.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, de sorte que la nouvelle valeur du masque est valide uniquement pendant la méthode actuelle. Voir la fonction de commande d'envoi pour la mise en œuvre des masques. Voir ABORT_ON_RESPONSE_CODE pour une liste des masques disponibles et leurs valeurs par défaut.

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

ABORT_ON_NO_DEVICE(VOID)

L'élément lexical du Builtin ABORT_ON_NO_DEVICE est présenté au Tableau B.13.

Tableau B.13 – Builtin ABORT_ON_NO_DEVICE

Nom du paramètre	Type	Direction	Usage	Description
<return>	VOID	—	—	—

B.14 Builtin ABORT_ON_RESPONSE_CODE

Objet

Le Builtin ABORT_ON_RESPONSE_CODE définit le(s) bit(s) correct(s) dans le masque d'interruption de code de réponse, de sorte que la valeur spécifiée du code de réponse provoque l'interruption de la méthode. Le code de réponse est défini comme étant le premier octet de données retourné dans une transaction, lorsque le bit 7 de cet octet est 0.

Les masques de relance et d'interruption sont réinitialisés à leurs valeurs par défaut au début de chaque méthode, de sorte que la nouvelle valeur du masque est valide uniquement pendant la méthode actuelle.

Une liste de tous les masques disponibles est présentée ci-dessous. Les valeurs par défaut des masques sont les suivantes:

- response_code_abort_mask = ACCESS_RESTRICTED | CMD_NOT_IMPLEMENTED
- response_code_retry_mask = NO BITS SET
- cmd_48_response_code_abort_mask = ACCESS_RESTRICTED | CMD_NOT_IMPLEMENTED
- cmd_48_response_code_retry_mask = NO BITS SET
- device_status_abort_mask = DEVICE_MALFUNCTION
- device_status_retry_mask = NO BITS SET
- cmd_48_device_status_abort_mask = DEVICE_MALFUNCTION
- cmd_48_device_status_retry_mask = NO BITS SET
- comm_status_abort_mask = NO BITS SET
- comm_status_retry_mask = ALL BITS SET
- cmd_48_comm_status_abort_mask = NO BITS SET
- cmd_48_comm_status_retry_mask = ALL BITS SET
- no_device_abort_mask = ALL BITS SET
- comm_error_abort_mask = ALL BITS SET
- comm_error_retry_mask = NO BITS SET
- cmd_48_no_device_abort_mask = ALL BITS SET
- cmd_48_comm_error_abort_mask = ALL BITS SET
- cmd_48_comm_error_retry_mask = NO BITS SET
- cmd_48_data_abort_mask = NO BITS SET
- cmd_48_data_retry_mask = NO BITS SET

Le masque affecté par cette fonction est utilisé par les Builtin send, send_trans, send_command, send_command_trans, ext_send_command, ext_send_command_trans, get_more_status et display.

Structure lexicale

```
void ABORT_ON_RESPONSE_CODE(response code)
```

Les éléments lexicaux du Builtin ABORT_ON_RESPONSE_CODE sont présentés au Tableau B.14.

Tableau B.14 – Builtin ABORT_ON_RESPONSE_CODE

Nom du paramètre	Type	Direction	Usage	Description
response_code	int	I	m	Spécifie un masque de bit pour le code de réponse
<return>	VOID	—	—	—

B.15 Builtin abort_on_response_code**Objet**

Le Builtin abort_on_response_code provoque l'interruption de la méthode à la réception du code de réponse spécifié.

Le Builtin provoque l'interruption de la méthode en cas de réception d'un code de réponse particulier. Ce Builtin évite qu'une méthode ne traite un code de réponse particulier.

Si une méthode est démarrée, elle ne s'interrompt pas sur un code de réponse particulier tant que ce Builtin n'est pas appelé. Un code de réponse est une valeur représentant une condition d'erreur spécifique à une application.

NOTE Une valeur de code de réponse zéro n'est pas considérée comme un code de réponse d'erreur. Chaque type de code de réponse est défini dans l'EDD.

Structure lexicale

```
abort_on_response_code(code)
```

Les éléments lexicaux du Builtin abort_on_response_code sont présentés au Tableau B.15.

Tableau B.15 – Builtin abort_on_response_code

Nom du paramètre	Type	Direction	Usage	Description
code	unsigned long	I	m	Spécifie un masque de bit pour le code de réponse
<return>	long	O	m	L'un des codes de retour spécifiés au Tableau B.235

B.16 Builtin abs**Objet**

Le Builtin abs renvoie la valeur absolue d'une valeur à virgule flottante.

Structure lexicale

```
abs(x)
```

Les éléments lexicaux du Builtin abs sont présentés au Tableau B.16.

Tableau B.16 – Builtin abs

Nom du paramètre	Type	Direction	Usage	Description
x	double	I	m	Valeur à virgule flottante
<return>	double	O	m	Valeur absolue