

# eFront Script Reference Guide - V21.0



# Table of Contents

<b>1. Welcome - Reference Guide .....</b>	<b>65</b>
<b>2. Overview .....</b>	<b>66</b>
2.1. What is eFront Script? .....	67
2.2. eFront - Reporting solutions and eFront Report .....	68
2.3. Example eFront Script code .....	70
2.4. Type of data being manipulated .....	72
2.5. Interdependency between steps and statements .....	73
<b>3. eFront Script syntax and notational conventions .....</b>	<b>75</b>
3.1. Notation of syntax elements .....	76
3.2. eFront Script program grammar & punctuation .....	77
3.3. Standard eFront Script data formats .....	78
3.3.1. Notes .....	78
3.4. eFront Script naming rules .....	79
3.4.1. Naming columns .....	79
3.4.2. Naming formats .....	79
3.4.3. Naming libraries .....	80
3.4.4. Naming eFront Cube cubes .....	80
3.4.5. Naming pictures .....	82
3.4.6. Naming tables .....	83
3.4.7. Naming transposed variables .....	83
3.4.8. Referencing files .....	84
3.4.9. Specifying paths .....	84
3.4.10. Specifying pictures .....	84
<b>4. eFront Script language elements .....</b>	<b>86</b>
4.1. Steps .....	87
4.1.1. DATA step .....	89
4.1.1.1. Description .....	89
4.1.1.2. Syntax .....	90
4.1.1.3. Subordinate statements .....	90
4.1.1.4. Comment .....	90
4.1.1.5. Output .....	91
4.1.1.6. Error messages .....	91
4.1.2. ODS step .....	91
4.1.2.1. Description .....	91

4.1.2.2. Syntax .....	91
4.1.2.3. Comment .....	91
4.1.2.4. Options .....	92
4.1.2.5. Output .....	92
4.1.2.6. Error messages .....	92
4.1.3. PROC CONVERTCURR step .....	92
4.1.3.1. Description .....	92
4.1.3.2. Syntax .....	92
4.1.3.3. Options .....	92
4.1.3.4. Subordinate Statements .....	93
4.1.3.5. Comment .....	93
4.1.3.6. Output .....	93
4.1.3.7. Error messages .....	94
4.1.4. PROC DELETE step .....	94
4.1.4.1. Description .....	94
4.1.4.2. Syntax .....	94
4.1.4.3. Options .....	94
4.1.4.4. Error messages .....	94
4.1.5. PROC DIRECTCASHFLOWS step .....	94
4.1.5.1. Description .....	94
4.1.5.2. Syntax .....	95
4.1.5.3. Options .....	95
4.1.5.4. Comment .....	96
4.1.5.5. Applicable macros .....	96
4.1.5.6. Output .....	96
4.1.5.7. Error messages .....	97
4.1.6. PROC EFRONTACCRUALS step .....	97
4.1.6.1. Description .....	97
4.1.6.2. Syntax .....	98
4.1.6.3. Options .....	98
4.1.6.4. Output .....	98
4.1.6.5. Error messages .....	98
4.1.7. PROC EFRONTBLOB step .....	99
4.1.7.1. Description .....	99
4.1.7.2. Syntax .....	99
4.1.7.3. Options for exporting groups BLOB fields .....	99
4.1.7.4. Options for exporting individual BLOB fields .....	100

4.1.7.5. Subordinate statements .....	101
4.1.7.6. Comment .....	101
4.1.7.7. Output .....	101
4.1.7.8. Error messages .....	101
4.1.8. PROC EFRONTDASHBOARD step .....	102
4.1.8.1. Description .....	102
4.1.8.2. Syntax .....	102
4.1.8.3. Options .....	102
4.1.8.4. Comment .....	103
4.1.8.5. Output .....	103
4.1.8.6. Error messages .....	103
4.1.9. PROC EFRONTFOLDER step .....	103
4.1.9.1. Description .....	103
4.1.9.2. Syntax .....	103
4.1.9.3. Options .....	103
4.1.9.4. Comment .....	105
4.1.9.5. Output .....	105
4.1.9.6. Error messages .....	105
4.1.10. PROC EFRONTIMPORT step .....	105
4.1.10.1. Description .....	105
4.1.10.2. Pre-requisite .....	105
4.1.10.3. Syntax .....	105
4.1.10.4. Options .....	105
4.1.10.5. Output .....	106
4.1.10.6. Error messages .....	106
4.1.11. PROC EFRONTIMPORT_AJX step .....	106
4.1.11.1. Description .....	106
4.1.11.2. Prerequisite .....	106
4.1.11.3. Syntax .....	106
4.1.11.4. Options .....	106
4.1.11.5. Comment .....	107
4.1.11.6. Output .....	107
4.1.11.7. Error messages .....	107
4.1.12. PROC EFRONTMAIL step .....	107
4.1.12.1. Description .....	107
4.1.12.2. Prerequisite .....	108
4.1.12.3. Syntax .....	108

4.1.12.4. Options .....	108
4.1.12.5. Output .....	109
4.1.12.6. Error messages .....	109
4.1.13. PROC EFRONTMENUS step .....	109
4.1.13.1. Description .....	109
4.1.13.2. Syntax .....	109
4.1.13.3. Options .....	109
4.1.13.4. Comment .....	109
4.1.13.5. Output .....	110
4.1.13.6. Error messages .....	110
4.1.14. PROC EFRONTPACKAGES step .....	110
4.1.14.1. Description .....	110
4.1.14.2. Syntax .....	111
4.1.14.3. Options .....	111
4.1.14.4. Output .....	112
4.1.14.5. Error messages .....	112
4.1.15. PROC EFRONTPROFILES step .....	113
4.1.15.1. Description .....	113
4.1.15.2. Syntax .....	113
4.1.15.3. Options .....	113
4.1.15.4. Output .....	113
4.1.15.5. Error messages .....	113
4.1.16. PROC EFRONTREGIONS step .....	114
4.1.16.1. Description .....	114
4.1.16.2. Syntax .....	114
4.1.16.3. Options .....	114
4.1.16.4. Output .....	115
4.1.16.5. Error messages .....	115
4.1.17. PROC EFRONTTABLES step .....	115
4.1.17.1. Description .....	115
4.1.17.2. Syntax .....	115
4.1.17.3. Options .....	115
4.1.17.4. Comment .....	116
4.1.17.5. Output .....	116
4.1.17.6. Error messages .....	116
4.1.18. PROC EFRONTUSERPROFILES step .....	116
4.1.18.1. Description .....	116

4.1.18.2. Syntax .....	116
4.1.18.3. Options .....	117
4.1.18.4. Output .....	117
4.1.18.5. Error messages .....	117
4.1.19. PROC ERROR step .....	117
4.1.19.1. Description .....	117
4.1.19.2. Syntax .....	117
4.1.19.3. Options .....	117
4.1.19.4. Comment .....	118
4.1.19.5. Output .....	118
4.1.19.6. Error messages .....	118
4.1.20. PROC EXPORTCHART step .....	118
4.1.20.1. Description .....	118
4.1.20.2. Syntax .....	118
4.1.20.3. Options .....	119
4.1.20.4. Comment .....	119
4.1.20.5. Output .....	119
4.1.21. PROC EXPORTEXCEL step .....	119
4.1.21.1. Description .....	119
4.1.21.2. Syntax .....	120
4.1.21.3. File options .....	120
4.1.21.4. Table options .....	120
4.1.21.5. Comment .....	121
4.1.21.6. Output .....	121
4.1.21.7. Error messages .....	121
4.1.22. PROC EXPORT step .....	121
4.1.22.1. Description .....	121
4.1.22.2. Syntax .....	122
4.1.22.3. Options .....	122
4.1.22.4. Comment .....	123
4.1.22.5. Output .....	123
4.1.22.6. Error messages .....	123
4.1.23. PROC FALIBRARY step .....	124
4.1.23.1. Description .....	124
4.1.23.2. Syntax to be used to call a standard library .....	124
4.1.23.3. OPTIONS to be used together with a call to a standard library .....	124

4.1.23.4. Parameters to be passed along with standard libraries ...	124
4.1.23.5. OPTIONS to be used together with a call to a specific library .....	125
4.1.23.6. Comment .....	126
4.1.23.7. Output .....	128
4.1.23.8. Error messages .....	128
4.1.24. PROC FAQUERY step .....	128
4.1.24.1. Description .....	128
4.1.24.2. Syntax .....	128
4.1.24.3. Options .....	128
4.1.24.4. ....	129
4.1.24.5. Subordinate statements .....	129
4.1.24.6. Comment .....	131
4.1.24.7. Output .....	131
4.1.24.8. Error messages .....	131
4.1.25. PROC FORMAT step .....	131
4.1.25.1. Description .....	131
4.1.25.2. Syntax .....	131
4.1.25.3. Options .....	132
4.1.25.4. Comment .....	132
4.1.25.5. Output .....	132
4.1.25.6. Error messages .....	132
4.1.26. PROC FUNDCASHFLOWS step .....	133
4.1.26.1. Description .....	133
4.1.26.2. Syntax .....	133
4.1.26.3. Options .....	133
4.1.26.4. Comment .....	134
4.1.26.5. Applicable macros .....	134
4.1.26.6. Output .....	135
4.1.26.7. Error messages .....	135
4.1.27. PROC GCHART step .....	135
4.1.27.1. Description .....	135
4.1.27.2. Syntax .....	135
4.1.27.3. Options .....	135
4.1.27.4. Subordinate statements .....	136
4.1.27.5. Output .....	136
4.1.27.6. Error messages .....	136

4.1.28. PROC GPLOT step .....	136
4.1.28.1. Description .....	136
4.1.28.2. Syntax .....	137
4.1.28.3. Options .....	137
4.1.28.4. Subordinate statements .....	137
4.1.28.5. Output .....	137
4.1.28.6. Error messages .....	137
4.1.29. PROC MEMORY step .....	138
4.1.29.1. Description .....	138
4.1.29.2. Syntax .....	138
4.1.29.3. Options .....	138
4.1.29.4. Comment .....	138
4.1.29.5. Output .....	138
4.1.29.6. Error messages .....	139
4.1.30. PROC MEANS step .....	139
4.1.30.1. Description .....	139
4.1.30.2. Syntax .....	139
4.1.30.3. Options .....	139
4.1.30.4. Subordinate statements and statistical functions .....	139
4.1.30.5. ....	140
4.1.30.6. Comment .....	140
4.1.30.7. Output on screen .....	140
4.1.30.8. Error messages .....	140
4.1.31. PROC OFFICE step .....	141
4.1.31.1. Description .....	141
4.1.31.2. Syntax .....	141
4.1.31.3. Options .....	141
4.1.31.4. Subordinate statements .....	141
4.1.31.5. ....	142
4.1.31.6. Comment .....	142
4.1.31.7. Output .....	142
4.1.31.8. Error messages .....	142
4.1.32. PROC PRINT step .....	142
4.1.32.1. Description .....	142
4.1.32.2. Syntax 1 .....	143
4.1.32.3. Syntax 2 .....	143
4.1.32.4. Options .....	143

4.1.32.5. Subordinate Statements .....	144
4.1.32.6. Embeddable steps .....	144
4.1.32.7. Comment .....	144
4.1.32.8. Output .....	145
4.1.32.9. Error messages .....	145
4.1.33. PROC PRINTCOL step .....	145
4.1.33.1. Description .....	145
4.1.33.2. Syntax .....	145
4.1.33.3. Options .....	145
4.1.33.4. Subordinate Statements .....	146
4.1.33.5. Comment .....	146
4.1.33.6. Output .....	146
4.1.33.7. Error messages .....	146
4.1.34. PROC PRINTFORM step .....	146
4.1.34.1. Description .....	146
4.1.34.2. Syntax .....	147
4.1.34.3. Options .....	147
4.1.34.4. Subordinate Statements .....	147
4.1.34.5. Comment .....	148
4.1.34.6. Output .....	148
4.1.34.7. Error messages .....	148
4.1.35. PROC SENDTOIC step .....	148
4.1.35.1. Description .....	148
4.1.35.2. Syntax .....	148
4.1.35.3. Options .....	149
4.1.35.4. Comment .....	150
4.1.35.5. Output .....	150
4.1.35.6. Error messages .....	151
4.1.36. PROC SETFILTER step .....	151
4.1.36.1. Description .....	151
4.1.36.2. Syntax .....	151
4.1.36.3. Options .....	151
4.1.36.4. Comment .....	152
4.1.36.5. Output .....	152
4.1.36.6. Error messages .....	152
4.1.37. PROC SORT step .....	152
4.1.37.1. Description .....	152

4.1.37.2. Syntax .....	152
4.1.37.3. Options .....	152
4.1.37.4. Subordinate statements .....	153
4.1.37.5. Comment .....	153
4.1.37.6. Output .....	153
4.1.37.7. Error messages .....	153
4.1.38. PROC SQLEXEC step .....	153
4.1.38.1. Description .....	153
4.1.38.2. Syntax .....	153
4.1.38.3. Options .....	154
4.1.38.4. Comment .....	154
4.1.38.5. Output .....	154
4.1.38.6. Error messages .....	154
4.1.39. PROC SQLIMPORT step .....	154
4.1.39.1. Description .....	154
4.1.39.2. Syntax .....	154
4.1.39.3. Options .....	155
4.1.39.4. Comment .....	156
4.1.39.5. Output .....	157
4.1.39.6. Error messages .....	157
4.1.40. PROC SQLTABLE step .....	157
4.1.40.1. Description .....	157
4.1.40.2. Syntax .....	157
4.1.40.3. Options .....	157
4.1.40.4. Comment .....	159
4.1.40.5. Output .....	159
4.1.40.6. Error messages .....	159
4.1.41. PROC TABULATE step .....	159
4.1.41.1. Description .....	159
4.1.41.2. Syntax .....	160
4.1.41.3. Options .....	160
4.1.41.4. Subordinate statements .....	160
4.1.41.5. Comment .....	161
4.1.41.6. Output .....	161
4.1.41.7. Error messages .....	161
4.1.42. PROC TRANSPOSE step .....	161
4.1.42.1. Description .....	161

4.1.42.2. Syntax .....	161
4.1.42.3. Output Options .....	161
4.1.42.4. Subordinate statements .....	162
4.1.42.5. Comment .....	162
4.1.42.6. Output .....	162
4.1.42.7. Error messages .....	162
4.1.43. PROC TRANSPOSE2 step .....	163
4.1.43.1. Description .....	163
4.1.43.2. Syntax .....	163
4.1.43.3. Options .....	163
4.1.43.4. Subordinate statements .....	163
4.1.43.5. Comment .....	163
4.1.43.6. Output .....	163
4.1.43.7. Error messages .....	164
4.1.44. PROC UPDATE step .....	164
4.1.44.1. Description .....	164
4.1.44.2. Syntax .....	164
4.1.44.3. Options .....	164
4.1.44.4. Subordinate statements .....	165
4.1.44.5. Comment .....	165
4.1.44.6. Output .....	165
4.1.44.7. PROC UPDATE and DATA MERGE - Comparison .....	165
4.1.44.8. Error messages .....	165
4.1.45. PROC WSGETCOMPANYDATA .....	166
4.1.45.1. Description .....	166
4.1.45.2. Syntax .....	166
4.1.45.3. Options .....	166
4.1.45.4. Comment .....	166
4.1.45.5. Output .....	166
4.1.45.6. Error messages .....	167
4.1.46. PROC WSGETINVESTMENTDATA .....	167
4.1.46.1. Description .....	167
4.1.46.2. Syntax .....	167
4.1.46.3. Options .....	167
4.1.46.4. Comment .....	168
4.1.46.5. Output .....	168
4.1.46.6. Error messages .....	168

4.1.47. PROC WSGETINVESTORSDATA .....	168
4.1.47.1. Description .....	168
4.1.47.2. Syntax .....	168
4.1.47.3. Options .....	169
4.1.47.4. Comment .....	169
4.1.47.5. Output .....	169
4.1.47.6. Error messages .....	169
4.1.48. STYLESHEET step .....	169
4.1.48.1. Description .....	169
4.1.48.2. Syntax .....	170
4.1.48.3. Style attributes .....	170
4.1.48.4. Comment .....	171
4.1.48.5. Output .....	171
4.1.48.6. Error messages .....	171
4.1.49. TABLE step .....	171
4.1.49.1. Description .....	171
4.1.49.2. EXAMPLE .....	172
4.1.49.3. Output .....	172
4.1.49.4. Error messages .....	172
4.2. Embeddable steps .....	173
4.2.1. EVENT step .....	173
4.2.1.1. Description .....	173
4.2.1.2. Syntax .....	173
4.2.1.3. Options .....	174
4.2.1.4. Comment .....	174
4.2.1.5. Output .....	174
4.2.1.6. Error messages .....	174
4.3. Global statements .....	175
4.3.1. LIBNAME statement .....	175
4.3.1.1. Description .....	175
4.3.1.2. Syntax .....	175
4.3.1.3. WORK Library Arguments .....	175
4.3.1.4. Comment .....	176
4.3.1.5. Error messages .....	176
4.3.1.6. DISK argument .....	176
4.3.1.7. MEMORY argument .....	177
4.4. Macro statements .....	179

4.4.1. %DEFINE statement .....	179
4.4.1.1. Description .....	179
4.4.1.2. Syntax .....	179
4.4.1.3. Comment .....	179
4.4.2. %INCLUDE statement .....	180
4.4.2.1. Description .....	180
4.4.2.2. Syntax .....	180
4.4.2.3. Options .....	180
4.4.2.4. Comment .....	180
4.4.2.5. Error messages .....	180
4.4.3. %CONTINUE statement .....	181
4.4.3.1. Description .....	181
4.4.3.2. Syntax .....	181
4.4.3.3. Comment .....	181
4.4.3.4. Error messages .....	181
4.4.4. %LET statement .....	181
4.4.4.1. Description .....	181
4.4.4.2. Syntax .....	181
4.4.4.3. Comment .....	182
4.4.5. %PARAM statement .....	182
4.4.5.1. Description .....	182
4.4.5.2. Syntax .....	182
4.4.5.3. Options .....	183
4.4.5.4. Comment .....	184
4.4.5.5. Error messages .....	184
4.4.6. %Param - Informats .....	184
4.5. Step statements .....	193
4.5.1. ABORT statement .....	195
4.5.1.1. Description .....	195
4.5.1.2. Syntax .....	195
4.5.1.3. Error messages .....	195
4.5.2. ARRAY statement .....	195
4.5.2.1. Description .....	195
4.5.2.2. Syntax .....	195
4.5.2.3. Comment .....	196
4.5.2.4. Error messages .....	196
4.5.3. BOX statement .....	196

4.5.3.1. Description .....	196
4.5.3.2. Syntax .....	196
4.5.3.3. Comment .....	197
4.5.3.4. Error messages .....	197
4.5.4. BY statement .....	197
4.5.4.1. Description .....	197
4.5.4.2. Syntax .....	197
4.5.4.3. Comment .....	197
4.5.4.4. Options .....	197
4.5.4.5. Error messages .....	199
4.5.5. CLASS statement .....	199
4.5.5.1. Description .....	199
4.5.5.2. Syntax .....	199
4.5.5.3. Comment .....	199
4.5.5.4. Error messages .....	200
4.5.6. COLUMN statement .....	200
4.5.6.1. Description .....	200
4.5.6.2. Syntax .....	200
4.5.6.3. Comment .....	200
4.5.6.4. Options .....	200
4.5.6.5. Error messages .....	201
4.5.7. CONTINUE statement .....	201
4.5.7.1. Description .....	201
4.5.7.2. Syntax .....	201
4.5.7.3. Comment .....	201
4.5.7.4. Error messages .....	201
4.5.8. DONUT statement .....	202
4.5.8.1. Description .....	202
4.5.8.2. Syntax .....	202
4.5.8.3. Options .....	202
4.5.8.4. Comment .....	202
4.5.8.5. Output .....	202
4.5.8.6. Error messages .....	202
4.5.9. DONUT3D statement .....	202
4.5.9.1. Description .....	202
4.5.9.2. Syntax .....	203
4.5.9.3. Options .....	203

4.5.9.4. Comment .....	203
4.5.9.5. Output .....	203
4.5.9.6. Error messages .....	203
4.5.10. EXTEND statement .....	203
4.5.10.1. Description .....	203
4.5.10.2. Syntax .....	204
4.5.10.3. Comment .....	204
4.5.10.4. Output .....	204
4.5.10.5. Error messages .....	204
4.5.11. EXPORT statement .....	204
4.5.11.1. Description .....	204
4.5.11.2. Syntax .....	204
4.5.11.3. Comment .....	205
4.5.11.4. Output .....	205
4.5.11.5. Error messages .....	205
4.5.12. FIRST statement .....	205
4.5.12.1. Description .....	205
4.5.12.2. Syntax .....	205
4.5.12.3. Options .....	205
4.5.12.4. Comment .....	206
4.5.12.5. Error messages .....	206
4.5.13. FORMAT statement .....	206
4.5.13.1. Description .....	206
4.5.13.2. Syntax .....	206
4.5.13.3. Comment .....	206
4.5.13.4. Error messages .....	206
4.5.14. HBAR statement .....	207
4.5.14.1. Description .....	207
4.5.14.2. Syntax .....	207
4.5.14.3. Options .....	207
4.5.14.4. Comment .....	207
4.5.14.5. Output .....	207
4.5.14.6. Error messages .....	207
4.5.15. HBAR3D statement .....	207
4.5.15.1. Description .....	207
4.5.15.2. Syntax .....	208
4.5.15.3. Options .....	208

4.5.15.4. Comment .....	208
4.5.15.5. Output .....	208
4.5.15.6. Error messages .....	208
4.5.16. ID statement .....	208
4.5.16.1. Description .....	208
4.5.16.2. Syntax .....	208
4.5.16.3. Comment .....	209
4.5.16.4. Error messages .....	209
4.5.17. IDLABEL statement .....	209
4.5.17.1. Description .....	209
4.5.17.2. Syntax .....	209
4.5.17.3. Comment .....	209
4.5.17.4. Error messages .....	209
4.5.18. PROC PRINT - SUM - STYLE .....	210
4.5.18.1. Goal .....	210
4.5.18.2. Features being illustrated .....	210
4.5.18.3. Program .....	210
4.5.18.4. Result .....	211
4.5.19. INFILE statement .....	211
4.5.19.1. Description .....	211
4.5.19.2. Syntax .....	211
4.5.19.3. Options .....	211
4.5.19.4. Comment .....	212
4.5.19.5. Error messages .....	212
4.5.20. INLINE statement .....	213
4.5.20.1. Description .....	213
4.5.20.2. Syntax .....	213
4.5.20.3. Comment .....	213
4.5.20.4. Error messages .....	213
4.5.21. IRR statement .....	214
4.5.21.1. Description .....	214
4.5.21.2. Syntax .....	214
4.5.21.3. Options .....	214
4.5.21.4. Comment .....	214
4.5.21.5. Error messages .....	215
4.5.22. JOIN statement .....	215
4.5.22.1. Description .....	215

4.5.22.2. Syntax .....	215
4.5.22.3. Comment .....	215
4.5.22.4. Error messages .....	215
4.5.23. LAST statement .....	216
4.5.23.1. Description .....	216
4.5.23.2. Syntax .....	216
4.5.23.3. Options .....	216
4.5.23.4. Comment .....	216
4.5.24. MAX statement .....	217
4.5.24.1. Description .....	217
4.5.24.2. Syntax .....	217
4.5.24.3. Options .....	217
4.5.24.4. Comment .....	217
4.5.24.5. Error messages .....	217
4.5.25. MEAN statement .....	217
4.5.25.1. Description .....	217
4.5.25.2. Syntax .....	218
4.5.25.3. Options .....	218
4.5.25.4. Comment .....	218
4.5.25.5. Error messages .....	218
4.5.26. MERGE statement .....	218
4.5.26.1. Description .....	218
4.5.26.2. Syntax .....	218
4.5.26.3. Comment .....	219
4.5.26.4. Example of the use of the IN flag together with the MERGE step .....	219
4.5.26.5. Error messages .....	220
4.5.27. MIN statement .....	220
4.5.27.1. Description .....	220
4.5.27.2. Syntax .....	220
4.5.27.3. Options .....	221
4.5.27.4. Comment .....	221
4.5.27.5. Error messages .....	221
4.5.28. MULTIPLE statement .....	221
4.5.28.1. Description .....	221
4.5.28.2. Syntax .....	221
4.5.28.3. Options .....	221

4.5.28.4. Comment .....	222
4.5.28.5. Error messages .....	222
4.5.29. N statement .....	222
4.5.29.1. Description .....	222
4.5.29.2. Syntax .....	222
4.5.29.3. Options .....	222
4.5.29.4. Comment .....	223
4.5.29.5. Error messages .....	223
4.5.30. NOBS statement .....	223
4.5.30.1. Description .....	223
4.5.30.2. Syntax .....	223
4.5.30.3. Comment .....	223
4.5.30.4. Output .....	224
4.5.30.5. Error messages .....	224
4.5.31. OUTPUT statement .....	224
4.5.31.1. Description .....	224
4.5.31.2. Syntax .....	224
4.5.31.3. Comment .....	224
4.5.31.4. Error messages .....	225
4.5.32. PICTURE statement .....	225
4.5.32.1. Description .....	225
4.5.32.2. Syntax .....	225
4.5.32.3. Comment .....	225
4.5.32.4. Error messages .....	225
4.5.33. PIE statement .....	226
4.5.33.1. Description .....	226
4.5.33.2. Syntax .....	226
4.5.33.3. Options .....	226
4.5.33.4. Comment .....	226
4.5.33.5. Output .....	226
4.5.33.6. Error messages .....	226
4.5.34. PIE3D statement .....	226
4.5.34.1. Description .....	226
4.5.34.2. Syntax .....	227
4.5.34.3. Options .....	227
4.5.34.4. Comment .....	227
4.5.34.5. Output .....	227

4.5.34.6. Error messages .....	227
4.5.35. PLOT statement .....	227
4.5.35.1. Description .....	227
4.5.35.2. Syntax .....	227
4.5.35.3. Options .....	228
4.5.35.4. Comment .....	228
4.5.35.5. Output .....	228
4.5.35.6. Error messages .....	228
4.5.36. PUT statement .....	228
4.5.36.1. Description .....	228
4.5.36.2. Syntax .....	228
4.5.36.3. Comment .....	229
4.5.36.4. Output .....	229
4.5.36.5. Error messages .....	229
4.5.37. RECURSE statement .....	229
4.5.37.1. Description .....	229
4.5.37.2. Syntax .....	229
4.5.37.3. Comment .....	229
4.5.37.4. Options .....	230
4.5.37.5. Error messages .....	230
4.5.38. RETURN statement .....	230
4.5.38.1. Description .....	230
4.5.38.2. Syntax .....	230
4.5.38.3. Comment .....	231
4.5.38.4. Error messages .....	231
4.5.39. SET statement .....	231
4.5.39.1. Description .....	231
4.5.39.2. Syntax .....	231
4.5.39.3. Comment .....	231
4.5.39.4. Error messages .....	232
4.5.40. SQL statement .....	232
4.5.40.1. Description .....	232
4.5.40.2. Syntax .....	232
4.5.40.3. Options .....	233
4.5.40.4. Comment .....	233
4.5.40.5. Error messages .....	233
4.5.41. STOP statement .....	233

4.5.41.1. Description .....	233
4.5.41.2. Syntax .....	233
4.5.41.3. Error messages .....	233
4.5.42. SUM statement .....	234
4.5.42.1. Description .....	234
4.5.42.2. Syntax .....	234
4.5.42.3. Options .....	234
4.5.42.4. Comment .....	234
4.5.42.5. Error messages .....	235
4.5.43. TABLE statement .....	235
4.5.43.1. Description .....	235
4.5.43.2. Syntax .....	235
4.5.43.3. Line structure - Column structure .....	235
4.5.43.4. Functions .....	236
4.5.43.5. Comment .....	236
4.5.43.6. Error messages .....	236
4.5.44. TEMPLATE statement .....	237
4.5.44.1. Description .....	237
4.5.44.2. Syntax .....	237
4.5.44.3. Comment .....	237
4.5.44.4. Error messages .....	237
4.5.45. TITLE statement .....	238
4.5.45.1. Description .....	238
4.5.45.2. Syntax .....	238
4.5.45.3. Comment .....	238
4.5.45.4. Error messages .....	238
4.5.46. TRACE statement .....	238
4.5.46.1. Description .....	238
4.5.46.2. Syntax .....	239
4.5.46.3. Comment .....	239
4.5.46.4. Error messages .....	239
4.5.47. TWR statement .....	239
4.5.47.1. Description .....	239
4.5.47.2. Syntax .....	239
4.5.47.3. Options .....	239
4.5.47.4. Comment .....	240
4.5.47.5. Error messages .....	243

4.5.48. VALUE statement .....	243
4.5.48.1. Description .....	243
4.5.48.2. Syntax .....	243
4.5.48.3. Comment .....	243
4.5.48.4. Error messages .....	244
4.5.49. VAR statement .....	244
4.5.49.1. Description .....	244
4.5.49.2. Syntax valid in a PROC TABULATE step .....	244
4.5.49.3. Syntax valid in a PROC MEANS step .....	244
4.5.49.4. Syntax valid in a PROC TRANSPOSE step .....	245
4.5.49.5. Syntax valid in a PROC PRINT step .....	245
4.5.49.6. Syntax valid in a PROC EFRONTBLOB step .....	245
4.5.49.7. Syntax valid in a PROC CONVERTCURR step .....	245
4.5.49.8. Options .....	246
4.5.49.9. Functions .....	246
4.5.49.10. Comment .....	246
4.5.49.11. Error messages .....	246
4.5.50. VBAR statement .....	247
4.5.50.1. Description .....	247
4.5.50.2. Syntax .....	247
4.5.50.3. Options .....	247
4.5.50.4. Comment .....	247
4.5.50.5. Output .....	247
4.5.50.6. Error messages .....	247
4.5.51. VBAR3D statement .....	247
4.5.51.1. Description .....	247
4.5.51.2. Syntax .....	248
4.5.51.3. Options .....	248
4.5.51.4. Comment .....	248
4.5.51.5. Output .....	248
4.5.51.6. Error messages .....	248
4.6. Options .....	249
4.6.1. Table options .....	249
4.6.1.1. Using KEEP .....	250
4.6.1.2. Using DROP .....	250
4.6.1.3. Using WHERE .....	250
4.6.1.4. Using IN .....	250

4.6.1.5. Using RENAME .....	251
4.6.1.6. Combining data filters .....	251
4.6.2. ALIGN option .....	251
4.6.2.1. Description .....	251
4.6.2.2. Syntax .....	251
4.6.2.3. Options .....	251
4.6.2.4. Comment .....	251
4.6.2.5. Error messages .....	251
4.6.3. FORMAT option .....	252
4.6.3.1. Description .....	252
4.6.3.2. Syntax .....	252
4.6.3.3. Comment .....	252
4.6.3.4. Error messages .....	252
4.6.4. FREEZEHEADER option .....	252
4.6.4.1. Description .....	252
4.6.4.2. Syntax .....	252
4.6.4.3. Comment .....	252
4.6.5. FREEZELEFT option .....	253
4.6.5.1. Description .....	253
4.6.5.2. Syntax .....	253
4.6.5.3. Comment .....	253
4.6.6. LABEL option .....	253
4.6.6.1. Description .....	253
4.6.6.2. Syntax - table option .....	253
4.6.6.3. Syntax - column option .....	253
4.6.6.4. Comment .....	253
4.6.6.5. Error messages .....	254
4.6.7. LENGTH option .....	254
4.6.7.1. Description .....	254
4.6.7.2. Syntax .....	254
4.6.7.3. Comment .....	254
4.6.7.4. Error messages .....	254
4.6.8. N option .....	254
4.6.8.1. Description .....	254
4.6.8.2. Syntax .....	255
4.6.8.3. Comment .....	255
4.6.8.4. Error messages .....	255

4.6.9. NAME option .....	255
4.6.9.1. Description .....	255
4.6.9.2. Syntax - column option .....	255
4.6.9.3. Error messages .....	256
4.6.10. NOGRANDTOTAL option .....	256
4.6.10.1. Description .....	256
4.6.10.2. Syntax .....	256
4.6.11. NODUPKEY option .....	256
4.6.11.1. Description .....	256
4.6.11.2. Syntax .....	256
4.6.11.3. Comment .....	256
4.6.11.4. Error messages .....	257
4.6.12. NOOBS option .....	257
4.6.12.1. Description .....	257
4.6.12.2. Syntax .....	257
4.6.12.3. Comment .....	257
4.6.12.4. Error messages .....	257
4.6.13. STYLE option .....	257
4.6.13.1. Description .....	257
4.6.13.2. Syntax - table option .....	258
4.6.13.3. Syntax - column option .....	258
4.6.13.4. Comment .....	259
4.6.13.5. Error messages .....	259
4.6.14. TYPE option .....	259
4.6.14.1. Description .....	259
4.6.14.2. Syntax .....	259
4.6.14.3. Comment .....	259
4.6.14.4. Types .....	259
4.6.14.5. Error messages .....	260
4.6.15. URL option .....	260
4.6.15.1. Description .....	260
4.6.15.2. Syntax .....	260
4.6.15.3. Options .....	260
4.6.16. WIDTH option .....	261
4.6.16.1. Description .....	261
4.6.16.2. Syntax .....	261
4.6.16.3. Comment .....	261

4.6.16.4. Error messages .....	261
4.7. Macro controls .....	262
4.7.1. %DO WHILE...%LOOP WHILE .....	262
4.7.1.1. Description .....	262
4.7.1.2. Syntax %DO WHILE .....	262
4.7.1.3. Syntax %LOOP WHILE .....	262
4.7.1.4. Comment .....	263
4.7.2. %FOR...%NEXT .....	263
4.7.2.1. Description .....	263
4.7.2.2. Syntax .....	263
4.7.2.3. Comment .....	263
4.7.2.4. Error messages .....	264
4.7.3. %IF...%THEN...%END .....	264
4.7.3.1. Description .....	264
4.7.3.2. Syntax .....	264
4.7.3.3. Comment .....	264
4.7.3.4. Error messages .....	265
4.7.4. %SELECT... %WHEN... %THEN... %END .....	265
4.7.4.1. Description .....	265
4.7.4.2. Syntax .....	265
4.7.4.3. Comment .....	266
4.7.4.4. Error messages .....	266
4.7.5. %WHILE...%END .....	266
4.7.5.1. Description .....	266
4.7.5.2. Syntax .....	266
4.7.5.3. Comment .....	267
4.7.6. Create a macro loop counter .....	267
4.8. Step controls .....	268
4.8.1. DO WHILE...LOOP WHILE .....	268
4.8.1.1. Description .....	268
4.8.1.2. Syntax DO WHILE .....	268
4.8.1.3. Syntax LOOP WHILE .....	269
4.8.1.4. Comment .....	269
4.8.1.5. Error messages .....	269
4.8.2. FOR...TO...NEXT .....	269
4.8.2.1. Description .....	269
4.8.2.2. Syntax .....	269

4.8.2.3. Comment .....	270
4.8.2.4. Error messages .....	270
4.8.3. FOR...IN...NEXT .....	270
4.8.3.1. Description .....	270
4.8.3.2. Syntax .....	270
4.8.3.3. Comment .....	271
4.8.3.4. Error messages .....	271
4.8.4. IF...THEN...END .....	271
4.8.4.1. Description .....	271
4.8.4.2. Syntax .....	271
4.8.4.3. Comment .....	272
4.8.4.4. Error messages .....	272
4.8.5. SELECT...WHEN...THEN...END .....	272
4.8.5.1. Description .....	272
4.8.5.2. Syntax .....	272
4.8.5.3. Comment .....	273
4.8.5.4. Error messages .....	274
4.8.6. Create a loop counter .....	274
4.9. Functions - Operators - Values .....	275
4.9.1. Functions - numbers .....	275
4.9.2. Functions - character strings .....	278
4.9.3. Functions - dates .....	281
4.9.4. Functions - files and folders .....	290
4.9.5. Functions - special .....	293
4.9.6. Operators .....	302
4.9.6.1. Comment .....	304
4.9.7. Special values .....	305
4.9.8. Specifying values or value ranges .....	305
4.10. Style attributes and values .....	307
4.10.1. Items to be styled .....	307
4.10.2. Chart style attributes and values .....	308
4.10.3. Text style attributes and values .....	312
4.11. Flags .....	314
4.11.1. _OUTPUT_flag .....	314
4.11.1.1. Description .....	314
4.11.1.2. Syntax .....	314
4.11.1.3. Comment .....	314

4.11.1.4. Error messages .....	315
4.11.2. IN flag .....	315
4.11.2.1. Description .....	315
4.11.2.2. Syntax .....	315
4.11.2.3. Comment .....	315
4.11.2.4. Error messages .....	316
4.12. Variables .....	317
4.12.1. Macro variables .....	317
4.12.1.1. Description .....	317
4.12.2. Magic variables .....	318
4.12.2.1. Description .....	318
4.12.2.2. Syntax .....	318
4.12.2.3. Error messages .....	319
4.12.3. The variable: ALL .....	319
4.12.3.1. Description .....	319
4.12.3.2. Syntax .....	319
4.12.3.3. Comment .....	319
4.12.3.4. Error messages .....	319
4.12.4. The variable: _LEVEL_ .....	320
4.12.4.1. Description .....	320
4.12.4.2. Syntax .....	320
4.12.4.3. Comment .....	320
4.12.4.4. Error messages .....	320
4.12.5. The variable: _URL_ .....	320
4.12.5.1. Description .....	320
4.12.5.2. Syntax 1 .....	320
4.12.5.3. Syntax 2 .....	321
4.12.5.4. Syntax 3 .....	321
4.12.5.5. Options .....	321
4.12.5.6. Comment .....	321
4.12.6. The variable: %DEBUG .....	321
4.12.6.1. Description .....	321
4.12.6.2. Syntax .....	321
4.12.6.3. Comment .....	322
4.13. XML reports .....	323
4.13.1. About XML reports .....	323
4.13.2. eFront XML Schema Definition .....	324

4.13.2.1. Goal .....	324
4.13.2.2. Specific eFront nodes that can be used .....	324
4.13.2.3. Comments .....	324
4.13.2.4. EF_ATTRIB .....	324
4.13.2.5. EF_G .....	325
4.13.2.6. EF_LOOP - EF_VALUE .....	326
4.13.2.7. EF_MACRO .....	327
4.13.2.8. EF_SELECT - EF_CASE - EF OTHERWISE .....	328
4.14. More about language elements .....	331
4.14.1. Understanding the DATA step .....	331
4.14.2. Understanding the SET statement .....	332
4.14.3. Understanding the MERGE statement .....	332
4.14.4. Understanding the TABLE statement .....	334
4.14.5. Understanding global statements .....	336
4.14.6. Understanding macro statements .....	336
4.14.7. Create import mask .....	337
4.14.8. Using styles .....	337
4.14.9. Using formats .....	338
<b>5. Examples - code snippets .....</b>	<b>339</b>
5.1. Examples - DATA step .....	342
5.1.1. DATA - ARRAY .....	342
5.1.1.1. Goal .....	342
5.1.1.2. Features being illustrated .....	343
5.1.1.3. Program .....	343
5.1.1.4. Result .....	346
5.1.2. DATA - COLUMN - OUTPUT .....	346
5.1.2.1. Goal .....	346
5.1.2.2. Features being illustrated .....	347
5.1.2.3. Program .....	347
5.1.2.4. Result .....	348
5.1.3. DATA - DO WHILE .....	348
5.1.3.1. Goal .....	348
5.1.3.2. Features being illustrated .....	348
5.1.3.3. Program .....	348
5.1.3.4. Result .....	349
5.1.4. DATA - DO WHILE - CONTINUE .....	349
5.1.4.1. Goal .....	349

5.1.4.2. Features being illustrated .....	349
5.1.4.3. Program .....	350
5.1.4.4. Result .....	351
5.1.5. DATA - DO WHILE - PUT .....	351
5.1.5.1. Goal .....	351
5.1.5.2. Features being illustrated .....	351
5.1.5.3. Program .....	351
5.1.5.4. Result .....	352
5.1.6. DATA - DO WHILE - PUT - HTML tags .....	352
5.1.6.1. Goal .....	352
5.1.6.2. Features being illustrated .....	353
5.1.6.3. Program .....	353
5.1.6.4. Result .....	354
5.1.7. DATA - EXTEND .....	354
5.1.7.1. Goal .....	354
5.1.7.2. Features being illustrated .....	354
5.1.7.3. Program .....	354
5.1.7.4. Result .....	355
5.1.8. DATA - FOR IN NEXT - %PARAM .....	355
5.1.8.1. Goal .....	355
5.1.8.2. Features being illustrated .....	355
5.1.8.3. Program .....	355
5.1.8.4. Result .....	357
5.1.9. DATA - FOR TO NEXT - OUTPUT .....	357
5.1.9.1. Goal .....	357
5.1.9.2. Features being illustrated .....	357
5.1.9.3. Program .....	357
5.1.9.4. Result .....	358
5.1.10. DATA - INFILE - COLUMN .....	359
5.1.10.1. Goal .....	359
5.1.10.2. Features being illustrated .....	359
5.1.10.3. Program .....	359
5.1.11. DATA - INFILE - INLINE (2) .....	360
5.1.11.1. Goal .....	360
5.1.11.2. Features being illustrated .....	360
5.1.11.3. Program .....	360
5.1.11.4. Result .....	362

5.1.12. DATA - INFILE - INLINE (1) .....	362
5.1.12.1. Goal .....	362
5.1.12.2. Features being illustrated .....	362
5.1.12.3. Program .....	362
5.1.12.4. Result .....	363
5.1.13. DATA - IF THEN ELSE .....	364
5.1.13.1. Goal .....	364
5.1.13.2. Features being illustrated .....	364
5.1.13.3. Program .....	364
5.1.13.4. Result .....	364
5.1.14. DATA - MERGE .....	365
5.1.14.1. Goal .....	365
5.1.14.2. Features being illustrated .....	365
5.1.14.3. Program .....	365
5.1.14.4. Result .....	366
5.1.15. DATA - MERGE - WHERE .....	366
5.1.15.1. Goal .....	366
5.1.15.2. Features being illustrated .....	366
5.1.15.3. Program .....	366
5.1.15.4. Result .....	367
5.1.15.5. Comment .....	367
5.1.16. DATA - MERGE - IN - IF THEN ELSE - OUTPUT .....	367
5.1.16.1. Goal .....	367
5.1.16.2. Features being illustrated .....	368
5.1.16.3. Program .....	368
5.1.16.4. Result .....	369
5.1.16.5. Comment .....	369
5.1.17. DATA - MERGE - IN - _OUTPUT_ .....	369
5.1.17.1. Goal .....	369
5.1.17.2. Features being illustrated .....	369
5.1.17.3. Program .....	369
5.1.17.4. Comment .....	370
5.1.18. DATA - OUTPUT - %PARAM .....	370
5.1.18.1. Goal .....	370
5.1.18.2. Features being illustrated .....	370
5.1.18.3. Program .....	370
5.1.19. DATA - RECURSE - IF THEN ELSE .....	371

5.1.19.1. Goal .....	371
5.1.19.2. Features being illustrated .....	371
5.1.19.3. Program .....	371
5.1.19.4. Result .....	372
5.1.20. DATA - RECURSE - IF THEN ELSE - _LEVEL_ .....	372
5.1.20.1. Goal .....	372
5.1.20.2. Features being illustrated .....	372
5.1.20.3. Program .....	373
5.1.20.4. Result .....	374
5.1.21. DATA - SET - WHERE .....	374
5.1.21.1. Goal .....	374
5.1.21.2. Features being illustrated .....	374
5.1.21.3. Program .....	374
5.1.21.4. Result .....	375
5.1.22. DATA - SET - SELECT WHEN THEN .....	375
5.1.22.1. Goal .....	375
5.1.22.2. Features being illustrated .....	375
5.1.22.3. Program .....	375
5.1.22.4. Result .....	377
5.1.23. DATA - SET - IN .....	377
5.1.23.1. Goal .....	377
5.1.23.2. Features being illustrated .....	377
5.1.23.3. Program .....	377
5.1.23.4. Result .....	378
5.1.24. DATA - SET - FCE.datatable .....	379
5.1.24.1. Goal .....	379
5.1.24.2. Features being illustrated .....	379
5.1.24.3. Program .....	379
5.1.25. DATA - SQL (1) .....	380
5.1.25.1. Goal .....	380
5.1.25.2. Features being illustrated .....	380
5.1.25.3. Program .....	380
5.1.25.4. Comment .....	380
5.1.26. DATA - SQL (2) .....	381
5.1.26.1. Goal .....	381
5.1.26.2. Features being illustrated .....	381
5.1.26.3. Program .....	381

5.1.26.4. Result .....	381
5.2. Examples - EVENT step .....	382
5.2.1. EVENT PREDATAROW - _URL_ .....	382
5.2.1.1. Goal .....	382
5.2.1.2. Features being illustrated .....	382
5.2.1.3. Program .....	382
5.2.1.4. Comment .....	383
5.2.1.5. Result .....	383
5.2.2. PROC PRINT - EVENT PREDATAROW .....	383
5.2.2.1. Goal .....	383
5.2.2.2. Features being illustrated .....	383
5.2.2.3. Program .....	383
5.2.2.4. Result .....	385
5.3. Examples - PROC CONVERTCURR step .....	386
5.3.1. PROC CONVERTCURR .....	386
5.3.1.1. Goal .....	386
5.3.1.2. Features being illustrated .....	386
5.3.1.3. Program .....	386
5.3.1.4. Result .....	389
5.4. Examples - PROC EFRONTACCRAULS step .....	390
5.4.1. PROC EFRONTACCRAULS - %PARAM .....	390
5.4.1.1. Goal .....	390
5.4.1.2. Features being illustrated .....	390
5.4.1.3. Program .....	390
5.4.1.4. Result .....	391
5.5. Examples - PROC EFRONTBLOB step .....	392
5.5.1. PROC EFRONTBLOB - %PARAM .....	392
5.5.1.1. Goal .....	392
5.5.1.2. Features being illustrated .....	392
5.5.1.3. Program .....	392
5.5.1.4. Result .....	393
5.5.2. PROC EFRONTBLOB - DATA - CLASS - PREFIX - ID .....	393
5.5.2.1. Goal .....	393
5.5.2.2. Features being illustrated .....	393
5.5.2.3. Program .....	394
5.5.2.4. Comment .....	394
5.6. Examples - PROC EFRONTDASHBOARD step .....	396

5.6.1. PROC EFRONTDASHBOARD - PAGES .....	396
5.6.1.1. Goal .....	396
5.6.1.2. Features being illustrated .....	396
5.6.1.3. Program .....	396
5.6.1.4. Result .....	396
5.6.1.5. Comment .....	396
5.6.2. PROC EFRONTDASHBOARD - dashboard parameters .....	396
5.6.2.1. Goal .....	396
5.6.2.2. Features being illustrated .....	397
5.6.2.3. Introduction .....	397
5.6.2.4. Program .....	397
5.6.2.5. Result .....	397
5.6.2.6. Comment .....	397
5.7. Examples - PROC EFRONTFOLDER step .....	398
5.7.1. PROC EFRONTFOLDER - FOLDER - ID - PROPERTIES .....	398
5.7.1.1. Goal .....	398
5.7.1.2. Features being illustrated .....	398
5.7.1.3. Program .....	398
5.7.1.4. Comment .....	398
5.7.2. PROC EFRONTFOLDER - Get portfolio positions .....	399
5.7.2.1. Goal .....	399
5.7.2.2. Features being illustrated .....	399
5.7.2.3. Program .....	400
5.7.2.4. Comment .....	400
5.8. Examples - PROC EFRONTIMPORT step .....	401
5.8.1. DATA - INFILE - PROC EFRONTIMPORT .....	401
5.8.1.1. Goal .....	401
5.8.1.2. Features being illustrated .....	401
5.8.1.3. Program .....	401
5.8.1.4. Comment .....	402
5.9. Examples - PROC EFRONTIMPORT_AJX step .....	403
5.9.1. PROC EFRONTIMPORT_AJX - %OUTPUTLOG .....	403
5.9.1.1. Goal .....	403
5.9.1.2. Features being illustrated .....	403
5.9.1.3. Program .....	403
5.9.1.4. Result .....	403
5.9.1.5. Comment .....	404

5.10. Examples - PROC EFRONTMENUS step .....	405
5.10.1. PROC EFRONTMENUS - GETUSERPROFILE - GETUSERID	405
5.10.1.1. Goal .....	405
5.10.1.2. Features being illustrated .....	405
5.10.1.3. Program .....	405
5.10.1.4. Result .....	406
5.11. Examples - PROC EFRONTTABLES step .....	407
5.11.1. PROC EFRONTTABLES - TABLES - INDEXES - RELATIONS	407
5.11.1.1. Goal .....	407
5.11.1.2. Features being illustrated .....	407
5.11.1.3. Program .....	407
5.11.1.4. Result .....	408
5.12. Examples - PROC EXPORT step .....	409
5.12.1. PROC EXPORT (1) .....	409
5.12.1.1. Goal .....	409
5.12.1.2. Features being illustrated .....	409
5.12.1.3. Program .....	409
5.12.1.4. Result .....	409
5.12.2. PROC EXPORT - FORMAT()	410
5.12.2.1. Goal .....	410
5.12.2.2. Features being illustrated .....	410
5.12.2.3. Program .....	410
5.12.2.4. Result .....	410
5.12.3. PROC EXPORT - %LET - PROC FORMAT .....	410
5.12.3.1. Goal .....	410
5.12.3.2. Features being illustrated .....	411
5.12.3.3. Program .....	411
5.12.3.4. Result .....	413
5.13. Examples - PROC EXPORTCHART step .....	414
5.13.1. PROC EXPORTCHART 1 - Generate graphs and integrate them into WORD docs .....	414
5.13.1.1. Goal .....	414
5.13.1.2. Features being illustrated .....	414
5.13.1.3. Algorithm used to implement example in eFront Script ...	414
5.13.1.4. Program .....	415
5.13.1.5. Result - Step 1: Excel Workbook with charts .....	418
5.13.1.6. WORD Template used to build the WORD report .....	419

5.13.1.7. Result - Final WORD report .....	421
5.13.1.8. ....	421
5.13.2. PROC EXPORTCHART - Generate graphs and integrate them into WORD docs .....	421
5.13.2.1. Goal .....	421
5.13.2.2. Features being illustrated .....	422
5.13.2.3. Program 1 .....	422
5.13.2.4. Result 1 .....	425
5.13.2.5. PROGRAM 2 .....	425
5.13.2.6. RESULT 2 .....	426
5.14. Examples - PROC EXPORTEXCEL step .....	430
5.14.1. PROC EXPORTEXCEL - GETTEMPPATH() .....	430
5.14.1.1. Goal .....	430
5.14.1.2. Features being illustrated .....	430
5.14.1.3. Program .....	430
5.15. Examples - PROC FALIBRARY step .....	431
5.15.1. Tip .....	431
5.15.2. PROC FALIBRARY - CFFPortfolio.PortfolioCashflows .....	434
5.15.2.1. Goal .....	434
5.15.2.2. Features being illustrated .....	435
5.15.2.3. Program .....	435
5.15.2.4. Comments .....	435
5.15.3. PROC FALIBRARY - HFFundAssets.Liquidity .....	438
5.15.3.1. Goal .....	438
5.15.3.2. Features being illustrated .....	439
5.15.3.3. Program .....	439
5.15.3.4. Comments .....	439
5.15.3.5. Parameters - How to get the internal names of library parameters .....	442
5.16. Examples - PROC FAQUERY step .....	446
5.16.1. PROC FAQUERY - Contacts - EntireTable .....	446
5.16.1.1. Goal .....	446
5.16.1.2. Features being illustrated .....	446
5.16.1.3. QueryBuilder query .....	446
5.16.1.4. Program .....	448
5.16.1.5. Result .....	449
5.16.1.6. ....	449

5.16.1.7. Comments .....	449
5.16.2. PROC FAQUERY - %LET .....	449
5.16.2.1. Goal .....	449
5.16.2.2. Features being illustrated .....	449
5.16.2.3. Program .....	449
5.16.2.4. Comments .....	450
5.17. Examples - PROC FORMAT step .....	451
5.17.1. PROC FORMAT - COLUMN - FORMAT .....	451
5.17.1.1. Goal .....	451
5.17.1.2. Features being illustrated .....	451
5.17.1.3. Program .....	451
5.17.1.4. Result .....	453
5.17.2. PROC FORMAT - PICTURE .....	453
5.17.2.1. Goal .....	453
5.17.2.2. Features being illustrated .....	453
5.17.2.3. Program .....	454
5.17.3. PROC FORMAT - VALUE (1) .....	454
5.17.3.1. Goal .....	454
5.17.3.2. Features being illustrated .....	454
5.17.3.3. Program .....	454
5.17.3.4. Result .....	456
5.17.4. PROC FORMAT - VALUE (2) .....	456
5.17.4.1. Goal .....	456
5.17.4.2. Features being illustrated .....	456
5.17.4.3. Program .....	456
5.17.4.4. Result .....	457
5.17.5. PROC FORMAT - PICTURE - large negative numbers .....	458
5.17.5.1. Goal .....	458
5.17.5.2. Features being illustrated .....	459
5.17.5.3. Program .....	459
5.18. Examples - PROC GCHART step .....	462
5.18.1. PROC GCHART - HBAR3D .....	462
5.18.1.1. Goal .....	462
5.18.1.2. Features being illustrated .....	462
5.18.1.3. Program .....	462
5.18.1.4. Result .....	463
5.19. Examples - PROC GPLOT step .....	464

5.19.1. PROC GPLOT - PLOT .....	464
5.19.1.1. Goal .....	464
5.19.1.2. Features being illustrated .....	464
5.19.1.3. Program .....	464
5.19.1.4. Result .....	465
5.20. Examples - PROC MEANS step .....	466
5.20.1. PROC MEANS - CLASS - N - LABEL .....	466
5.20.1.1. Goal .....	466
5.20.1.2. Features being illustrated .....	466
5.20.1.3. Program .....	466
5.20.1.4. Result .....	467
5.20.2. PROC MEANS - JOIN .....	467
5.20.2.1. Goal .....	467
5.20.2.2. Features being illustrated .....	467
5.20.2.3. Program .....	467
5.20.2.4. Result .....	469
5.20.3. PROC MEANS - JOIN - FIRST - LAST .....	469
5.20.3.1. Goal .....	469
5.20.3.2. Features being illustrated .....	469
5.20.3.3. Program .....	469
5.20.3.4. Result .....	471
5.20.4. PROC MEANS - LEFT() - N .....	471
5.20.4.1. Goal .....	471
5.20.4.2. Features being illustrated .....	471
5.20.4.3. Program .....	471
5.20.4.4. Result .....	472
5.20.5. PROC MEANS - MEAN - SUM - SQRT() .....	472
5.20.5.1. Goal .....	472
5.20.5.2. Features being illustrated .....	473
5.20.5.3. Program .....	473
5.20.6. PROC MEANS - MISSING .....	475
5.20.6.1. Goal .....	475
5.20.6.2. Features being illustrated .....	475
5.20.6.3. Program .....	475
5.20.6.4. Result .....	477
5.20.7. PROC MEANS - CLASS - NOBS .....	477
5.20.7.1. Goal .....	477

5.20.7.2. Features being illustrated .....	478
5.20.7.3. Program .....	478
5.20.7.4. Result .....	479
5.20.8. PROC MEANS - PROC SORT - MAX .....	479
5.20.8.1. Goal .....	479
5.20.8.2. Features being illustrated .....	479
5.20.8.3. Program .....	479
5.20.8.4. Comment .....	480
5.20.9. PROC MEANS - SUM .....	480
5.20.9.1. Goal .....	480
5.20.9.2. Features being illustrated .....	480
5.20.9.3. Program .....	480
5.20.9.4. Result .....	481
5.20.10. PROC MEANS - SUM - MEAN - N .....	481
5.20.10.1. Goal .....	481
5.20.10.2. Features being illustrated .....	481
5.20.10.3. Program .....	482
5.20.10.4. Result .....	483
5.20.11. PROC MEANS - SUM - SEMESTER() .....	483
5.20.11.1. Goal .....	483
5.20.11.2. Features being illustrated .....	483
5.20.11.3. Program .....	483
5.20.11.4. Comment .....	484
5.20.11.5. Result .....	484
5.20.12. PROC MEANS - IRR .....	485
5.20.12.1. Goal .....	485
5.20.12.2. Features being illustrated .....	485
5.20.12.3. Program .....	485
5.20.12.4. Comment .....	486
5.20.12.5. Result .....	486
5.20.13. PROC MEANS - VAR - MULTIPLE .....	487
5.20.13.1. Goal .....	487
5.20.13.2. Features being illustrated .....	487
5.20.13.3. Program .....	487
5.20.13.4. Comment .....	488
5.21. Examples - PROC OFFICE step .....	489
5.21.1. PROC OFFICE - Generate XML report .....	489

5.21.1.1. Goal .....	489
5.21.1.2. Features being illustrated .....	489
5.21.1.3. Program .....	489
5.21.1.4. Template XML file used to produce the XML report .....	490
5.21.1.5. Tables being used to feed the XML report .....	491
5.21.1.6. Result .....	492
5.21.2. PROC OFFICE - Generate multi WORD document generator ...	492
5.21.2.1. Goal .....	492
5.21.2.2. Features being illustrated .....	493
5.21.2.3. Program .....	493
5.22. Examples - PROC PRINT step .....	495
5.22.1. Example - IRR CASH FLOWS BY INSTRUMENT .....	495
5.22.1.1. Goal .....	495
5.22.1.2. Features being illustrated .....	496
5.22.1.3. Program .....	497
5.22.2. PROC PRINT - BY - COLLAPSE - EXPAND .....	498
5.22.2.1. Goal .....	498
5.22.2.2. Features being illustrated .....	498
5.22.2.3. Program .....	498
5.22.2.4. Result .....	499
5.22.2.5. Comment .....	499
5.22.3. PROC PRINT - LABEL .....	499
5.22.3.1. Goal .....	499
5.22.3.2. Features being illustrated .....	499
5.22.3.3. Program .....	499
5.22.3.4. Result .....	500
5.22.4. PROC PRINT - NOOBS - N .....	500
5.22.4.1. Goal .....	500
5.22.4.2. Features being illustrated .....	500
5.22.4.3. Program .....	500
5.22.4.4. Result .....	501
5.22.5. PROC PRINT - N - FORMAT - SUM - BY .....	501
5.22.5.1. Goal .....	501
5.22.5.2. Features being illustrated .....	501
5.22.5.3. Program .....	501
5.22.5.4. Result .....	502
5.22.6. PROC PRINT - PUT .....	503

5.22.6.1. Goal .....	503
5.22.6.2. Features being illustrated .....	503
5.22.6.3. Program .....	503
5.22.6.4. Result .....	503
5.22.7. PROC PRINT - PUT-- Investor Summary (Report + Chart) .....	503
5.22.7.1. Goal .....	503
5.22.7.2. Features being illustrated .....	504
5.22.7.3. Program .....	504
5.22.8. PROC PRINT - SUM - STYLE .....	507
5.22.8.1. Goal .....	507
5.22.8.2. Features being illustrated .....	507
5.22.8.3. Program .....	507
5.22.8.4. Result .....	508
5.22.9. PROC PRINT - SUM - BY - STYLE .....	508
5.22.9.1. Goal .....	508
5.22.9.2. Features being illustrated .....	508
5.22.9.3. Program .....	509
5.22.9.4. Result .....	509
5.22.10. PROC PRINT - URL .....	510
5.22.10.1. Goal .....	510
5.22.10.2. Features being illustrated .....	510
5.22.10.3. Program .....	510
5.22.10.4. Comment .....	510
5.22.11. PROC PRINT - VAR - .SUM - STYLE .....	511
5.22.11.1. Goal .....	511
5.22.11.2. Features being illustrated .....	511
5.22.11.3. Program .....	511
5.22.11.4. Result .....	512
5.22.11.5. Comment .....	512
5.22.12. PROC PRINT - VAR - .MULTIPLE .....	512
5.22.12.1. Goal .....	512
5.22.12.2. Features being illustrated .....	512
5.22.12.3. Program .....	512
5.22.12.4. Comment .....	513
5.23. Examples - PROC PRINTCOL step .....	514
5.23.1. PROC PRINTCOL - FLOW - STYLESHEET .....	514
5.23.1.1. Goal .....	514

5.23.1.2. Features being illustrated .....	514
5.23.1.3. Program .....	514
5.23.1.4. Result .....	516
5.24. Examples - PROC PRINTFORM step .....	517
5.24.1. PROC PRINTFORM - TITLE - VAR .....	517
5.24.1.1. Goal .....	517
5.24.1.2. Features being illustrated .....	517
5.24.1.3. Program .....	517
5.24.1.4. Result .....	518
5.24.2. PROC PRINTFORM - TABLENOBS() - %IF %THEN %END ....	519
5.24.2.1. Goal .....	519
5.24.2.2. Features being illustrated .....	519
5.24.2.3. Program .....	519
5.24.2.4. COMMENT .....	519
5.25. Examples - PROC SENDTOIC step .....	520
5.25.1. PROC SENDTOIC .....	520
5.25.1.1. Goal .....	520
5.25.1.2. Features being illustrated .....	520
5.25.1.3. Program .....	520
5.26. Examples - PROC SORT step .....	522
5.26.1. PROC SORT - NODUPKEY - BY .....	522
5.26.1.1. Goal .....	522
5.26.1.2. Features being illustrated .....	522
5.26.1.3. Program .....	522
5.26.1.4. Result .....	523
5.26.2. PROC SORT - BY - UPCASE .....	523
5.26.2.1. Goal .....	523
5.26.2.2. Features being illustrated .....	523
5.26.2.3. Program .....	523
5.26.2.4. Result .....	524
5.27. Examples - PROC SQLIMPORT step .....	526
5.27.1. PROC SQLIMPORT - TABLE - FILTER .....	526
5.27.1.1. Goal .....	526
5.27.1.2. Features being illustrated .....	526
5.27.1.3. Program .....	526
5.27.1.4. Comment .....	526
5.27.2. PROC SQLIMPORT - CONNECTION - SQL - MAXROWS .....	526

5.27.2.1. Goal .....	526
5.27.2.2. Features being illustrated .....	527
5.27.2.3. Program .....	527
5.27.2.4. Comment .....	527
5.27.3. PROC SQLIMPORT - SQL - MAXROWS - ??FILTER .....	527
5.27.3.1. Goal .....	527
5.27.3.2. Features being illustrated .....	527
5.27.3.3. Program .....	527
5.27.3.4. Comment .....	528
5.27.4. PROC SQLIMPORT - VIEW .....	528
5.27.4.1. Goal .....	528
5.27.4.2. Features being illustrated .....	528
5.27.4.3. Program .....	528
5.27.4.4. Comment .....	528
5.27.5. PROC SQLIMPORT - COLUMNMAPPING .....	528
5.27.5.1. Goal .....	528
5.27.5.2. Features being illustrated .....	529
5.27.5.3. Program .....	529
5.27.5.4. Result .....	530
5.27.5.5. Comment .....	530
5.28. Examples - PROC SQLTABLE step .....	531
5.28.1. PROC SQLTABLE - (1) .....	531
5.28.1.1. Goal .....	531
5.28.1.2. Features being illustrated .....	531
5.28.1.3. Program .....	531
5.28.1.4. Comment .....	531
5.28.2. PROC SQLTABLE - (2) .....	531
5.28.2.1. Goal .....	531
5.28.2.2. Features being illustrated .....	532
5.28.2.3. Program .....	532
5.28.2.4. Result .....	533
5.28.2.5. Extract of eFront.config .....	533
5.28.2.6. Comment .....	533
5.28.3. PROC SQLTABLE - COLUMNMAPPING .....	534
5.28.3.1. Goal .....	534
5.28.3.2. Features being illustrated .....	534
5.28.3.3. Program .....	534

5.28.3.4. Result .....	536
5.28.3.5. Comment .....	536
5.29. Examples - PROC TABULATE step .....	537
5.29.1. PROC TABULATE - BOX - STYLE .....	537
5.29.1.1. Goal .....	537
5.29.1.2. Features being illustrated .....	537
5.29.1.3. Program .....	537
5.29.1.4. Result .....	538
5.29.2. PROC TABULATE - TABLE - ALL - N - SUM .....	538
5.29.2.1. Goal .....	538
5.29.2.2. Features being illustrated .....	538
5.29.2.3. Program .....	538
5.29.2.4. Result .....	539
5.29.3. PROC TABULATE - TABLE - STYLE (1) .....	539
5.29.3.1. Features being illustrated .....	539
5.29.3.2. Program .....	539
5.29.3.3. Result .....	540
5.29.4. PROC TABULATE - TABLE - STYLE (2) .....	540
5.29.4.1. Goal .....	540
5.29.4.2. Features being illustrated .....	541
5.29.4.3. Program .....	541
5.29.4.4. Result .....	542
5.29.5. PROC TABULATE - TITLE .....	542
5.29.5.1. Goal .....	542
5.29.5.2. Features being illustrated .....	542
5.29.5.3. Program .....	542
5.29.5.4. Result .....	543
5.30. Examples - PROC TRANSPOSE step .....	544
5.30.1. PROC TRANSPOSE (1) .....	544
5.30.1.1. Goal .....	544
5.30.1.2. Features being illustrated .....	544
5.30.1.3. Program .....	544
5.30.1.4. Result .....	545
5.30.1.5. Comment .....	545
5.30.2. PROC TRANSPOSE (2) .....	545
5.30.2.1. Goal .....	545
5.30.2.2. Features being illustrated .....	545

5.30.2.3. Program .....	545
5.30.2.4. Result .....	546
5.30.2.5. Comment .....	546
5.31. Examples - PROC TRANSPOSE2 step .....	548
5.31.1. PROC TRANSPOSE2 - MultiCurrency Transactions .....	548
5.31.1.1. Goal .....	548
5.31.1.2. Features being illustrated .....	548
5.31.1.3. Program .....	548
5.31.1.4. Result .....	551
5.31.1.5. Comment .....	551
5.31.2. PROC TRANSPOSE - SingleCurrency Transactions .....	551
5.31.2.1. Goal .....	551
5.31.2.2. Features being illustrated .....	552
5.31.2.3. Program .....	552
5.31.2.4. Result .....	553
5.31.2.5. Comment .....	554
5.32. Examples - PROC UPDATE .....	555
5.32.1. PROC UPDATE - Merge two cubes .....	555
5.32.1.1. Goal .....	555
5.32.1.2. Features being illustrated .....	555
5.32.1.3. Program .....	555
5.32.1.4. Result before the merge .....	556
5.32.1.5. Result after the merge .....	557
5.32.1.6. Comment .....	557
5.32.2. PROC UPDATE and DATA MERGE - Comparison .....	558
5.32.2.1. Features being illustrated .....	558
5.32.2.2. Program .....	558
5.32.2.3. Results before the merge .....	561
5.32.2.4. .....	561
5.32.2.5. Results after the merge .....	561
5.32.2.6. .....	561
5.33. Examples - STYLESHEET step .....	562
5.33.1. STYLESHEET - PROC TABULATE (1) .....	562
5.33.1.1. Goal .....	562
5.33.1.2. Features being illustrated .....	562
5.33.1.3. Program .....	562
5.33.1.4. Result .....	564

5.33.2. STYLESHEET - PROC TABULATE (2) .....	564
5.33.2.1. Goal .....	564
5.33.2.2. Features being illustrated .....	564
5.33.2.3. Program .....	565
5.33.2.4. Result .....	567
5.33.2.5. Comment .....	567
5.33.3. STYLESHEET - PROC PRINT .....	567
5.33.3.1. Goal .....	567
5.33.3.2. Features being illustrated .....	567
5.33.3.3. Program .....	567
5.33.3.4. Result .....	569
5.33.3.5. .....	569
5.33.3.6. Comment .....	569
5.34. Examples - global statements .....	570
5.34.1. LIBNAME - Magic variables .....	570
5.34.1.1. Goal .....	570
5.34.1.2. Features being illustrated .....	570
5.34.1.3. Program .....	570
5.34.1.4. Comment .....	571
5.34.2. LIBNAME - USER - CURRENT .....	571
5.34.2.1. Goal .....	571
5.34.2.2. Features being illustrated .....	571
5.34.2.3. Program .....	571
5.34.2.4. .....	571
5.34.2.5. Comment .....	571
5.35. Examples - macro statements and controls .....	572
5.35.1. %DEFINE .....	572
5.35.1.1. Goal .....	572
5.35.1.2. Features being illustrated .....	572
5.35.1.3. Program .....	573
5.35.1.4. Result .....	573
5.35.2. %FOR - %NEXT - MOD .....	573
5.35.2.1. Goal .....	573
5.35.2.2. Features being illustrated .....	574
5.35.2.3. Program .....	574
5.35.2.4. Comment .....	574
5.35.2.5. Result .....	575

5.35.3. %INCLUDE .....	575
5.35.3.1. Goal .....	575
5.35.3.2. Features being illustrated .....	575
5.35.3.3. Program .....	575
5.35.3.4. Result .....	576
5.35.4. %INCLUDE - %PARAM - %IF %THEN %END .....	576
5.35.4.1. Goal .....	576
5.35.4.2. Features being illustrated .....	577
5.35.4.3. Program .....	577
5.35.4.4. Result .....	577
5.35.5. %LET - %DO WHILE - %LOOP WHILE .....	578
5.35.5.1. Goal .....	578
5.35.5.2. Features being illustrated .....	578
5.35.5.3. Program .....	578
5.35.5.4. Result .....	579
5.35.6. %LET - %FOR - %NEXT .....	580
5.35.6.1. Goal .....	580
5.35.6.2. Features being illustrated .....	580
5.35.6.3. Program .....	580
5.35.6.4. Result .....	581
5.35.7. %LET - %FOR - %NEXT - @TABLE .....	581
5.35.7.1. Goal .....	581
5.35.7.2. Features being illustrated .....	581
5.35.7.3. Program .....	581
5.35.7.4. Comment .....	582
5.35.7.5. Result .....	583
5.35.8. %LET - %SELECT - %WHEN - %END .....	583
5.35.8.1. Goal .....	583
5.35.8.2. Features being illustrated .....	583
5.35.8.3. Program .....	583
5.35.8.4. Result .....	585
5.35.9. %LET - %WHILE - %END .....	585
5.35.9.1. Goal .....	585
5.35.9.2. Features being illustrated .....	585
5.35.9.3. Program .....	585
5.35.9.4. Result .....	586
5.35.10. %LET - INSTR() .....	586

5.35.10.1. Goal .....	586
5.35.10.2. Features being illustrated .....	586
5.35.10.3. Program .....	586
5.35.10.4. Comment .....	587
5.35.11. %LET - LINKFILES() - LINKFILE() .....	587
5.35.11.1. Goal .....	587
5.35.11.2. Features being illustrated .....	588
5.35.11.3. Program .....	588
5.35.11.4. Comment .....	589
5.35.12. %PARAM .....	589
5.35.12.1. Goal .....	589
5.35.12.2. Features being illustrated .....	589
5.35.12.3. Program .....	589
5.35.12.4. Comment .....	590
5.35.13. %PARAM - ??CHOICE - LIKE .....	590
5.35.13.1. Goal .....	590
5.35.13.2. Features being illustrated .....	590
5.35.13.3. Program .....	590
5.35.13.4. Comment .....	591
5.35.14. %PARAM - ??ID - WHERE .....	591
5.35.14.1. Goal .....	591
5.35.14.2. Features being illustrated .....	591
5.35.14.3. Program .....	591
5.35.14.4. Comment .....	592
5.35.15. %PARAM - LINKFILESEXT() - LINKFILE() .....	592
5.35.15.1. Goal .....	592
5.35.15.2. Features being illustrated .....	593
5.35.15.3. Program .....	593
5.35.15.4. Comment .....	593
5.35.16. %PARAM - TYPE - DEFAULT .....	593
5.35.16.1. Goal .....	593
5.35.16.2. Features being illustrated .....	594
5.35.16.3. Program .....	594
5.35.16.4. Result .....	594
5.35.17. %PARAM - ??LOOKUP - ??XLOOKUPCODE .....	594
5.35.17.1. Goal .....	594
5.35.17.2. Features being illustrated .....	595

5.35.17.3. Program .....	595
5.35.17.4. Result .....	595
5.35.18. %PARAM - ??PICKID(SQL- query) - ??FILTER .....	596
5.35.18.1. Goal .....	596
5.35.18.2. Features being illustrated .....	596
5.35.18.3. Program .....	596
5.35.19. %PARAM - ??PICKID_{VAR} .....	596
5.35.19.1. Goal .....	596
5.35.19.2. Features being illustrated .....	596
5.35.19.3. Program .....	596
5.35.19.4. Comment .....	597
5.35.19.5. Result .....	597
5.35.20. %PARAM - ??XPICKID - (?) - (?) .....	597
5.35.20.1. Goal .....	597
5.35.20.2. Features being illustrated .....	598
5.35.20.3. Program .....	598
5.35.20.4. Comment .....	598
5.35.20.5. Result .....	599
5.35.21. %PARAM - ??XPICKID - SHOWSELECTEDITEMS .....	599
5.35.21.1. Goal .....	599
5.35.21.2. Features being illustrated .....	599
5.35.21.3. Program .....	600
5.35.21.4. Comment .....	600
5.35.22. %PARAM - ??XPICKID with SQL query .....	600
5.35.22.1. Goal .....	600
5.35.22.2. Features being illustrated .....	600
5.35.22.3. Program .....	600
5.36. Examples - functions and operators .....	601
5.36.1. DATEADD() - FORMAT() .....	601
5.36.1.1. Goal .....	601
5.36.1.2. Features being illustrated .....	601
5.36.1.3. Program .....	602
5.36.1.4. Result .....	602
5.36.2. DATEDIFF() - FORMAT() .....	603
5.36.2.1. Goal .....	603
5.36.2.2. Features being illustrated .....	603
5.36.2.3. Program .....	603

5.36.2.4. Result .....	604
5.36.3. DATEDIFF() - SEMESTERENDDATE() - 1 .....	604
5.36.3.1. Goal .....	604
5.36.3.2. Features being illustrated .....	604
5.36.3.3. Program .....	604
5.36.3.4. Result .....	605
5.36.4. DATEDIFF() - SEMESTERENDDATE() - 2 .....	605
5.36.4.1. Goal .....	605
5.36.4.2. Features being illustrated .....	606
5.36.4.3. Program .....	606
5.36.4.4. Result .....	606
5.36.5. DATEDIFF() - QuarterBegDate() - DateADD() .....	607
5.36.5.1. Goal .....	607
5.36.5.2. Features being illustrated .....	607
5.36.5.3. Program .....	607
5.36.5.4. Result .....	609
5.36.5.5. Comments .....	609
5.36.6. DMY() - MDY() - FORMAT() .....	609
5.36.6.1. Goal .....	609
5.36.6.2. Features being illustrated .....	609
5.36.6.3. Program .....	609
5.36.6.4. Result .....	610
5.36.7. FILENAME() - FILE() .....	610
5.36.7.1. Goal .....	610
5.36.7.2. Features being illustrated .....	611
5.36.7.3. Program .....	611
5.36.7.4. Comment .....	612
5.36.7.5. Result .....	612
5.36.8. GETTEMPPATH() .....	612
5.36.8.1. Goal .....	612
5.36.8.2. Features being illustrated .....	612
5.36.8.3. Program .....	612
5.36.8.4. Comment .....	613
5.36.9. GETUSERID()_GETUSERINFO() .....	614
5.36.9.1. Goal .....	614
5.36.9.2. Features being illustrated .....	614
5.36.9.3. Program .....	614

5.36.9.4. Available fields .....	615
5.36.9.5. Comment .....	615
5.36.9.6. Result .....	615
5.36.10. LINKFILES() - LINKFILE() - FILENAME() .....	615
5.36.10.1. Goal .....	615
5.36.10.2. Features being illustrated .....	615
5.36.10.3. Program .....	616
5.36.10.4. Comment .....	616
5.36.11. LINKFILES() - LINKFILE() - FILENAME() - %PARAM .....	616
5.36.11.1. Goal .....	616
5.36.11.2. Features being illustrated .....	616
5.36.11.3. Program .....	616
5.36.12. LINKFILES() - LINKFILE() - PROC MEMORY - Macro controls .....	617
5.36.12.1. Goal .....	617
5.36.12.2. Features being illustrated .....	617
5.36.12.3. Program .....	617
5.36.12.4. Result .....	621
5.36.12.5. EXCEL Report Template .....	622
5.36.13. LOOKUP()_LOOKUPCODE() .....	623
5.36.13.1. Goal .....	623
5.36.13.2. Features being illustrated .....	623
5.36.13.3. Prerequisite .....	623
5.36.13.4. Program .....	625
5.36.13.5. Result .....	626
5.36.14. PROC SORT - (WHERE - NOT...IN) .....	626
5.36.14.1. Goal .....	626
5.36.14.2. Features being illustrated .....	626
5.36.14.3. Program .....	626
5.36.15. ROUND()_CINT() .....	626
5.36.15.1. Goal .....	626
5.36.15.2. Features being illustrated .....	627
5.36.15.3. Program .....	627
5.36.15.4. Result .....	628
5.36.16. WEEKDAY() .....	628
5.36.16.1. Goal .....	628
5.36.16.2. Features being illustrated .....	628

5.36.16.3. Comment .....	628
5.36.16.4. Program .....	629
5.36.16.5. Result - EXCEL report .....	630
5.36.17. YMD() .....	630
5.36.17.1. Goal .....	630
5.36.17.2. Features being illustrated .....	630
5.36.17.3. Program .....	630
5.36.17.4. Result .....	631
5.36.18. YMD() - FORMAT() .....	632
5.36.18.1. Goal .....	632
5.36.18.2. Features being illustrated .....	632
5.36.18.3. Program .....	632
5.36.18.4. Result .....	633
5.37. Examples - performance increase for eFront Cube programs .....	634
5.37.1. PROC SETFILTER - EXTEND - Global Cube and Filtered Session Cube .....	634
5.37.1.1. Goal .....	634
5.37.1.2. Features being illustrated .....	634
5.37.1.3. 1. Program A - Build the global cube .....	634
5.37.1.4. 1. Result - Program A .....	635
5.37.1.5. 2. Program B - Build the session cube that filters the global cube .....	637
5.37.1.6. 2. Result: Program B .....	638
5.37.1.7. 3. Link cubes and program to dashboard .....	639
5.38. Examples - XML Schema Definition .....	644
5.38.1. EF_LOOP-EF_VALUE-EF_ATTRIB-EF_G-EF_MACRO .....	644
5.38.1.1. Goal .....	644
5.38.1.2. Features being illustrated .....	644
5.38.1.3. XML Template .....	644
5.38.1.4. XML File being built with the Template .....	645
5.38.2. EF_SELECT-EF_CASE-EF OTHERWISE .....	646
5.38.2.1. Goal .....	646
5.38.2.2. Features being illustrated .....	646
5.38.2.3. XML Template .....	646
5.38.2.4. XML File being built with the Template .....	647
<b>6. eFront Script - Calling the Calculation Engine .....</b>	<b>649</b>

6.1. About eFront Invest calculation engines	650
About eFront Invest calculation engines .....	650
6.2. Accessing CE tables from within an eFront Script program .....	651
6.3. Data table description .....	652
6.4. Examples .....	653
6.5. About eFront Invest calculation engines .....	654
6.6. Integrating CE tables into eFront Script programs .....	656
6.6.1. Available tables - DCE .....	656
6.6.2. Available tables - FCE .....	656
6.6.3. Available tables - CCE .....	657
6.6.4. Integration of CE tables into eFront Script program .....	657
6.6.5. Examples .....	657
6.7. Macros and variables to be used when calling CE tables .....	658
6.7.1. Examples of macro usage .....	658
6.7.2. Macros and variables to be used when calling DCE tables .....	659
6.7.2.1. Applicable macros .....	659
6.7.2.2. Examples of macro usage .....	661
6.7.3. Macros and variables to be used when calling FCE tables .....	661
6.7.3.1. Applicable macros .....	661
6.7.3.2. Examples of macro usage .....	664
6.7.4. Macros and variables to be used when calling CCE tables .....	664
6.7.4.1. Applicable macros .....	664
6.7.4.2. Examples of macro usage .....	664
6.8. Data tables produced by Direct CE (DCE) .....	665
6.8.1. DirectCashFlows .....	665
6.8.1.1. DIRECTCASHFLOWS .....	665
6.8.2. InstrumentPositions .....	672
6.8.2.1. INSTRUMENTPOSITIONS .....	672
6.8.3. RefInstrumentPositions .....	680
6.8.3.1. REFINSTRUMENTPOSITIONS .....	680
6.8.4. InvestmentPositions .....	686
6.8.4.1. INVESTMENTPOSITIONS .....	686
6.8.5. InvestorPositions .....	692
6.8.5.1. INVESTORPOSITIONS .....	692
6.8.6. InstrumentRounds .....	698
6.8.6.1. INSTRUMENTROUNDS .....	698
6.8.7. CompanyPositions .....	705

6.8.7.1. COMPANYPOSITIONS .....	705
6.8.8. FV_T_PORTCO_TRANSACTIONS .....	709
6.8.8.1. FV_T_PORTCO_TRANSACTIONS, FV_T_CASH_TRANSACTIONS, FV_T_FEES_TRANSACTIONS	709
6.8.9. FV_T_PORTCO_POSITIONS_BY_SECURITIES .....	719
6.8.9.1. FV_T_PORTCO_POSITIONS_BY_SECURITIES, FV_T_CASH_POSITIONS_BY_SECURITIES, FV_T_FEES_POSITIONS_BY_SECURITIES .....	719
6.8.10. FV_T_PORTCO_POSITIONS_BY_INVESTMENT .....	729
6.8.10.1. FV_T_PORTCO_POSITIONS_BY_INVESTMENTS,FV_T_CASH _POSITIONS_BY_INVESTMENTS,FV_T_FEES_POSITIONS_B Y_INVESTMENTS .....	729
6.8.11. FV_T_PORTCO_POSITIONS_BY_INVESTORS .....	737
6.8.11.1. FV_T_PORTCO_POSITIONS_BY_INVESTORS,FV_T_CASH_P OSITIONS_BY_INVESTORS,FV_T_FEES_POSITIONS_BY_INV ESTORS .....	737
6.9. Data tables produced by the Funds CE (FCE) .....	742
6.9.1. FundOperations .....	779
6.9.1.1. FUNDOPERATIONS .....	779
6.9.2. FundShareOperations .....	782
6.9.2.1. FUNDSHAREOPERATIONS .....	782
6.9.3. FundInvestorOperations .....	785
6.9.3.1. FUNDINVESTOROPERATIONS .....	785
6.9.4. FundInvestorShareOperations .....	788
6.9.4.1. FUNDINVESTORSHAREOPERATIONS .....	788
6.9.5. FundPositions and FundPositionsWithoutCarried .....	791
6.9.5.1. FUNDPOSITIONS / FUNDPOSITIONSWITHOUTCARRIED .....	791
6.9.6. FundInvestorPositions .....	794
6.9.6.1. FUNDINVESTORPOSITIONS .....	794
6.9.7. FundSharePositions .....	798
6.9.7.1. FUNDSHAREPOSITIONS .....	798
6.9.8. FundInvestorSharePositions .....	801
6.9.8.1. FUNDINVESTORSHAREPOSITIONS .....	801
6.9.9. FundCashFlows .....	805

6.9.9.1. FUND CASHFLOWS .....	805
6.9.10. InvestorPositions .....	808
6.9.10.1. INVESTORPOSITIONS .....	808
6.9.11. MasterfundInvestorPositions .....	811
6.9.11.1. MASTERFUNDINVESTORPOSITIONS .....	811
6.9.12. FV_T_FUND_TRANSACTIONS .....	814
6.9.12.1. FV_T_FUND_TRANSACTIONS .....	814
6.9.13. FV_T_FUND_SEC_POSITIONS_BY_INVESTMENTS .....	823
6.9.13.1. FV_T_FUND_SEC_POSITIONS_BY_INVESTMENTS ..	823
6.9.14. FV_T_FUND_SEC_POSITIONS_BY_FUNDS .....	831
6.9.14.1. FV_T_FUND_SEC_POSITIONS_BY_FUNDS .....	831
6.9.15. FV_T_FUND_POSITIONS_BY_FUNDS .....	839
6.9.15.1. FV_T_FUND_POSITIONS_BY_FUNDS .....	839
6.9.16. FV_T_FUND_POSITIONS_BY_INVESTMENTS .....	843
6.9.16.1. FV_T_FUND_POSITIONS_BY_FUNDS .....	843
6.9.17. FV_T_FUND_POSITIONS_BY_INVESTORS .....	853
6.9.17.1. FV_T_FUND_POSITIONS_BY_INVESTORS .....	853
6.9.18. FV_T_FUND_OPERATIONS_BY_INVESTORS .....	856
6.9.18.1. FV_T_FUND_OPERATIONS_BY_INVESTORS .....	856
6.9.19. FV_T_FUND_OPERATIONS_BY_FUNDS .....	860
6.9.19.1. FV_T_FUND_OPERATIONS_BY_FUNDS .....	860
6.10. Data tables produced by the Consolidated CE (CCE) .....	865
6.10.1. InvestmentPositions .....	867
6.10.1.1. INVESTMENTPOSITIONS .....	867
6.10.2. InvestorPositions .....	869
6.10.2.1. INVESTORPOSITIONS .....	869
6.11. Details about calculations .....	872
6.12. Examples - Calculation Engine Tables .....	873
6.12.1. Goal .....	873
6.12.2. Features being illustrated .....	873
6.12.3. Program .....	873
6.12.4. Goal .....	875
6.12.5. Features being illustrated .....	875
6.12.6. Program .....	876
6.12.7. Goal .....	877
6.12.8. Features being illustrated .....	877
6.12.9. Program .....	877

6.12.10. DATA - SET - CCE.datatable .....	879
6.12.10.1. Goal .....	879
6.12.10.2. Features being illustrated .....	879
6.12.10.3. Program .....	879
6.12.11. DATA - SET - DCE.datatable .....	881
6.12.11.1. Goal .....	881
6.12.11.2. Features being illustrated .....	881
6.12.11.3. Program .....	881
6.12.12. DATA - SET - FCE.datatable .....	883
6.12.12.1. Goal .....	883
6.12.12.2. Features being illustrated .....	883
6.12.12.3. Program .....	884
<b>7. Managing systematic access to external databases .....</b>	<b>885</b>
7.1. Define access to an external database .....	886
7.2. Accessing eFront Data Warehouse data using eFront Script .....	887
7.2.1. Configuring access to the eFront Data Warehouse .....	887
7.2.2. Importing and exporting eFront Data Warehouse data .....	887
7.2.3. Filtering eFront Data Warehouse data .....	888
7.2.3.1. Notes .....	889
<b>8. Error debugging .....</b>	<b>890</b>
8.1. Error messages .....	891
8.1.1. '=' expected .....	891
8.1.2. 0001 DATA_NONE .....	891
8.1.3. Cannot find Office template file .....	891
8.1.4. Cannot mix column and statistic at same level .....	891
8.1.5. Cannot register table .....	891
8.1.6. Column defined more than once in a format clause .....	891
8.1.7. column is not defined in table .....	891
8.1.8. ERREUR: Option de variable invalide : .....	892
8.1.9. ERROR(0105): {EOF} expected .....	892
8.1.10. ERROR(0106): '(' expected .....	892
8.1.11. ERROR(0107): ')' expected .....	892
8.1.12. ERROR(0111): ';' expected .....	892
8.1.13. ERROR(0113): {Column name} expected .....	892
8.1.14. ERROR(0113): {Format name} expected .....	892
8.1.15. ERROR(0113): {prefix} expected .....	892
8.1.16. ERROR(0113): 'RUN' expected .....	892

8.1.17. ERROR(0113): Stylesheet name expected .....	893
8.1.18. ERROR(0113): Table name expected .....	893
8.1.19. ERROR(0113): 'THEN' expected .....	893
8.1.20. ERROR(0113): Variable name expected .....	893
8.1.21. ERROR(0121): {filename} expected .....	893
8.1.22. ERROR(0121): {string} expected .....	893
8.1.23. ERROR(0121): {template-name} expected .....	893
8.1.24. ERROR(0123): Missing function parameter()s .....	894
8.1.25. ERROR(0123): Too many function parameters .....	894
8.1.26. ERROR(0127): TOP value expected .....	894
8.1.27. ERROR(0130): '.' expected .....	894
8.1.28. ERROR: Chart type not defined .....	894
8.1.29. ERROR: Column name defined twice .....	894
8.1.30. ERROR: Format defined twice .....	894
8.1.31. ERROR: Invalid definition .....	894
8.1.32. ERROR: Invalid proc option .....	895
8.1.33. ERROR: Invalid style element .....	895
8.1.34. ERROR: Invalid style parameter .....	895
8.1.35. ERROR: Missing or duplicate column name .....	895
8.1.36. ERROR: No BY column defined .....	895
8.1.37. invalid PROC qualifier .....	895
8.1.38. ERROR: Syntax error in data options definition! .....	895
8.1.39. ERROR: WHEN, OTHERWISE or END expected .....	895
8.1.40. Format not found .....	896
8.1.41. Include object not found .....	896
8.1.42. 'INTEGER' expected after LENGTH .....	896
8.1.43. Invalid ALIGN value .....	896
8.1.44. Invalid name .....	896
8.1.45. Invalid Office template .....	896
8.1.46. Invalid or missing COLUMN macro parameter. ....	896
8.1.47. Invalid or unknown library .....	896
8.1.48. Invalid path or path not found .....	897
8.1.49. No calc variable defined .....	897
8.1.50. Invalid TYPE value .....	897
8.1.51. Invalid value in inline data .....	897
8.1.52. Office template file is not defined .....	897
8.1.53. Stylesheet not found .....	897

8.1.54. 'TO' expected .....	897
8.1.55. Unknown BY column .....	897
8.1.56. Unknown CLASS column .....	898
8.1.57. Unknown IRR column .....	898
8.1.58. Unknown MAX column .....	898
8.1.59. Unknown MEAN column .....	898
8.1.60. Unknown MIN column .....	898
8.1.61. Unknown MULTIPLE column .....	898
8.1.62. Unknown N column .....	898
8.1.63. Unknown name .....	898
8.1.64. Unknown SUM column .....	898
8.1.65. Unknown TABLE column .....	899
<b>9. Dictionary of &lt;FrontScript&gt; tokens .....</b>	<b>900</b>
9.1. % .....	901
9.2. %CONTINUE .....	902
9.3. %DO .....	903
9.4. %ELSE .....	904
9.5. %ELSEIF .....	905
9.6. %END .....	906
9.7. %EXIT %DO .....	907
9.8. %EXIT %FOR .....	908
9.9. %FOR .....	909
9.10. %IF .....	910
9.11. %INCLUDE .....	911
9.12. %LET .....	912
9.13. %LOOP .....	913
9.14. %NEXT .....	914
9.15. %PARAM .....	915
9.16. %THEN .....	916
9.17. %TO .....	917
9.18. %WHILE .....	918
9.19. ??CHECK .....	919
9.20. ??CHOICE .....	920
9.21. ??FILTER .....	921
9.22. ??ID .....	922
9.23. ??LOOKUP .....	923
9.24. ??LOOKUPCODE .....	924

9.25. ??PICKID .....	925
9.26. ??XID .....	926
9.27. ??XLOOKUP .....	927
9.28. ??XLOOKUPCODE .....	928
9.29. ??XPICKID .....	929
9.30. _LEVEL_ .....	930
9.31. _OUTPUT_ .....	931
9.32. _URL_ .....	932
9.33. ABORT .....	933
9.34. ALIGN .....	934
9.35. ALL .....	935
9.36. AMOUNT .....	936
9.37. AND .....	937
9.38. ATTACHMENTS .....	938
9.39. BANNER .....	939
9.40. BANNERFILE .....	940
9.41. BCC .....	941
9.42. BETWEEN .....	942
9.43. BLOB .....	943
9.44. BODY .....	944
9.45. BOOLEAN .....	945
9.46. BOX .....	946
9.47. BY .....	947
9.48. BYTE .....	948
9.49. CALCULATE SHARES .....	949
9.50. CC .....	950
9.51. CCE .....	951
9.52. CE_CURRENCY .....	952
9.53. CHAR .....	953
9.54. CLASS .....	954
9.55. CLOSING .....	955
9.56. COLHEADER .....	956
9.57. COLLAPSE .....	957
9.58. COLUMN .....	958
9.59. COLUMNS .....	959
9.60. COMMA .....	960
9.61. CONTINUE .....	961

9.62. DATA .....	962
9.63. DATE .....	963
9.64. DCE .....	964
9.65. DEFAULT .....	965
9.66. DELIMITER .....	966
9.67. DESCENDING .....	967
9.68. DETAILS .....	968
9.69. DIRECTCASHFLOWS .....	969
9.70. DO .....	970
9.71. DONUT .....	971
9.72. DONUT3D .....	972
9.73. DOUBLE .....	973
9.74. DRIVER .....	974
9.75. DROP .....	975
9.76. EF_ATTRIB .....	976
9.77. EF_CASE .....	977
9.78. EF_G .....	978
9.79. EF_LOOP .....	979
9.80. EF_MACRO .....	980
9.81. EF OTHERWISE .....	981
9.82. EF_SELECT .....	982
9.83. EF_SELECT .....	983
9.84. EF_VALUE .....	984
9.85. EFRONTACCRLS .....	985
9.86. EFRONTBLOB .....	986
9.87. EFRONTFOLDER .....	987
9.88. EFRONTIMPORT .....	988
9.89. EFRONTIMPORT_AJX .....	989
9.90. EFRONTMAIL .....	990
9.91. EFRONTMENUS .....	991
9.92. EFRONTPACKAGES .....	992
9.93. EFRONTPROFILES .....	993
9.94. EFRONTREGIONS .....	994
9.95. EFRONTTABLES .....	995
9.96. ELSE .....	996
9.97. ELSEIF .....	997
9.98. EMPTY .....	998

9.99. END .....	999
9.100. EVENT .....	1000
9.101. EXCEL .....	1001
9.102. EXIT DO .....	1002
9.103. EXIT FOR .....	1003
9.104. EXPAND .....	1004
9.105. EXPORT .....	1005
9.106. EXPORTEXCEL .....	1006
9.107. EXTEND .....	1007
9.108. FALIBRARY .....	1008
9.109. FALSE .....	1009
9.110. FAQUERY .....	1010
9.111. FCE .....	1011
9.112. FILE .....	1012
9.113. FILES .....	1013
9.114. FILTER .....	1014
9.115. FIRST .....	1015
9.116. FIRSTOBS .....	1016
9.117. FLOAT .....	1017
9.118. FLOW .....	1018
9.119. FOR .....	1019
9.120. FORMAT .....	1020
9.121. FREEZEHEADER .....	1021
9.122. FREEZELEFT .....	1022
9.123. FROM .....	1023
9.124. FUNDCASHFLOWS .....	1024
9.125. GCHART .....	1025
9.126. GPLOT .....	1026
9.127. GRANDTOTAL .....	1027
9.128. GROUPFORMAT .....	1028
9.129. HBAR .....	1029
9.130. HBAR3D .....	1030
9.131. HEADER .....	1031
9.132. HORIZONTALLY .....	1032
9.133. HTML .....	1033
9.134. HTMLBODY .....	1034
9.135. ID .....	1035

9.136. IDLABEL .....	1036
9.137. IF .....	1037
9.138. IN .....	1038
9.139. INDEXES .....	1039
9.140. INFILE .....	1040
9.141. INFO .....	1041
9.142. INFORMAT .....	1042
9.143. INLINE .....	1043
9.144. INSTRUMENTS .....	1044
9.145. INTEGER .....	1045
9.146. INTERPOL .....	1046
9.147. IRR .....	1047
9.148. IS .....	1048
9.149. JOIN .....	1049
9.150. KEEP .....	1050
9.151. LABEL .....	1051
9.152. LAST .....	1052
9.153. LASTOBS .....	1053
9.154. LENGTH .....	1054
9.155. LIB .....	1055
9.156. LIBNAME .....	1056
9.157. LIBRARY .....	1057
9.158. LIKE .....	1058
9.159. LISTING .....	1059
9.160. LOGICAL .....	1060
9.161. LONG .....	1061
9.162. LOOKUP .....	1062
9.163. LOOKUPCODE .....	1063
9.164. LOOP .....	1064
9.165. MAX .....	1065
9.166. MEAN .....	1066
9.167. MEANS .....	1067
9.168. MEMORY .....	1068
9.169. MERGE .....	1069
9.170. MIN .....	1070
9.171. MISSING .....	1071
9.172. MOD .....	1072

9.173. MULTIPLE .....	1073
9.174. N .....	1074
9.175. NAME .....	1075
9.176. NBOBS .....	1076
9.177. NEXT .....	1077
9.178. NODATA .....	1078
9.179. NODUPKEY .....	1079
9.180. NOERROR .....	1080
9.181. NOGRANDTOTAL .....	1081
9.182. NOLIST .....	1082
9.183. NOOBS .....	1083
9.184. NORELOAD .....	1084
9.185. NOT .....	1085
9.186. NOTHING .....	1086
9.187. NOTNULL .....	1087
9.188. NULL .....	1088
9.189. OBS .....	1089
9.190. ODS .....	1090
9.191. OFFICE .....	1091
9.192. OR .....	1092
9.193. OTHERWISE .....	1093
9.194. OUT .....	1094
9.195. OUTPUT .....	1095
9.196. PARENT .....	1096
9.197. PATH .....	1097
9.198. PCTN .....	1098
9.199. PCTSUM .....	1099
9.200. PICTURE .....	1100
9.201. PIE .....	1101
9.202. PIE3D .....	1102
9.203. PLOT .....	1103
9.204. PREDATAROW .....	1104
9.205. PREFIX .....	1105
9.206. PRINT .....	1106
9.207. PRINTCOL .....	1107
9.208. PRINTFORM .....	1108
9.209. PRIVATE .....	1109

9.210. PROC .....	1110
9.211. PUBLIC .....	1111
9.212. PUT .....	1112
9.213. REAL .....	1113
9.214. RECURSE .....	1114
9.215. RELATIONS .....	1115
9.216. RENAME .....	1116
9.217. REPORT_DATE .....	1117
9.218. RETURN .....	1118
9.219. RUN .....	1119
9.220. SELECT .....	1120
9.221. SEMESTER .....	1121
9.222. SEMESTERBEGDATE .....	1122
9.223. SEMESTERENDDATE .....	1123
9.224. SEMICOLON .....	1124
9.225. SENDTOIC .....	1125
9.226. SET .....	1126
9.227. SETFILTER .....	1127
9.228. SHARED .....	1128
9.229. SHOWSELECTEDITEMS .....	1129
9.230. SORT .....	1130
9.231. SORTDESC .....	1131
9.232. SPACE .....	1132
9.233. SQL .....	1133
9.234. SQLEXEC .....	1134
9.235. SQLIMPORT .....	1135
9.236. SQLTABLE .....	1136
9.237. START .....	1137
9.238. STOP .....	1138
9.239. STRING .....	1139
9.240. STYLE .....	1140
9.241. STYLESHEET .....	1141
9.242. SUBJECT .....	1142
9.243. SUBSCRFUNDOP .....	1143
9.244. SUBSCRFUNDOPSHARE .....	1144
9.245. SUM .....	1145
9.246. SUMVAR .....	1146

9.247. SYMBOL .....	1147
9.248. TAB .....	1148
9.249. TABLE .....	1149
9.250. TABLENOBS .....	1150
9.251. TABLES .....	1151
9.252. TABULATE .....	1152
9.253. TEMPLATE .....	1153
9.254. TEXT .....	1154
9.255. THEN .....	1155
9.256. TIME .....	1156
9.257. TITLE .....	1157
9.258. TITLE1 .....	1158
9.259. TITLE10 .....	1159
9.260. TITLE2 .....	1160
9.261. TITLE3 .....	1161
9.262. TITLE4 .....	1162
9.263. TITLE5 .....	1163
9.264. TITLE6 .....	1164
9.265. TITLE7 .....	1165
9.266. TITLE8 .....	1166
9.267. TITLE9 .....	1167
9.268. TO .....	1168
9.269. TO .....	1169
9.270. TOP .....	1170
9.271. TRACE .....	1171
9.272. TRANSACTIONS .....	1172
9.273. TRANSPOSE .....	1173
9.274. TRUE .....	1174
9.275. TYPE .....	1175
9.276. UNLOAD .....	1176
9.277. UPCASE .....	1177
9.278. UPDATE .....	1178
9.279. URL .....	1179
9.280. USE_DRAFT .....	1180
9.281. USE_INVESTEE_CURR .....	1181
9.282. USE_INVESTOR_CURR .....	1182
9.283. VALUE .....	1183

9.284. VAR .....	1184
9.285. VBAR .....	1185
9.286. VBAR3D .....	1186
9.287. VERTICALLY .....	1187
9.288. WHEN .....	1188
9.289. WHERE .....	1189
9.290. WHILE .....	1190
9.291. WIDTH .....	1191
9.292. WITH .....	1192

## 1. Welcome - Reference Guide

Welcome to the **eFront Script Reference Guide**.

This document is available as online help. Please note that the information contained within the document can be easily accessed from the following locations:

- **Contents**: use this tool to get a general idea of the information being available
- **Index**: use this tool to get information related to a particular topic, such as 'labels, display labels as column titles'.
- **Search**: use this tool to get information related to a precise **eFront Script** keyword, such as COLUMN.

If you need complementary support, please contact your eFront consultant.

## 2. Overview

The **eFront Script Reference Guide** covers **eFront Script** basic language elements, providing syntax references, comments, and example code snippets. It is intended for anyone who needs to generate reports, charts and listings using the **eFront Script** programming language.

- [What is eFront Script?](#)
- [What is eFront Report for?](#)
- [Example eFront Script code](#)
- [Type of data being manipulated](#)
- [Interdependency between steps and statements](#)

## 2.1. What is eFront Script?

**eFront Script** is the **eFront** programming language used to write data process programs. Data process programs have two main objectives:

- Extract data from databases (ex.: eFront Invest Classic, eFront Invest, external databases), and build **eFront Report** data tables or **eFront Analytics data cubes**.
- Process data tables to calculate statistics, and produce tabular reports, listings, charts, dashboards, **WORD** and **EXCEL** documents.

**eFront Script** is a database-oriented fourth-generation programming language (4GL) developed for data manipulation, data analysis and reporting.

**eFront Script** is a **procedural programming language**. Each **eFront Script** program is composed of DATA steps that build the data tables, and a series of procedure steps that process the data tables and generate reports.

The DATA step section of an **eFront Script** program assumes a default file structure, and automates the process of identifying files to the operating system, opening the input file, reading the next record, opening the output file, writing the next record, and closing the files. Most of the time the DATA step works within an implicit program loop that runs for each record.

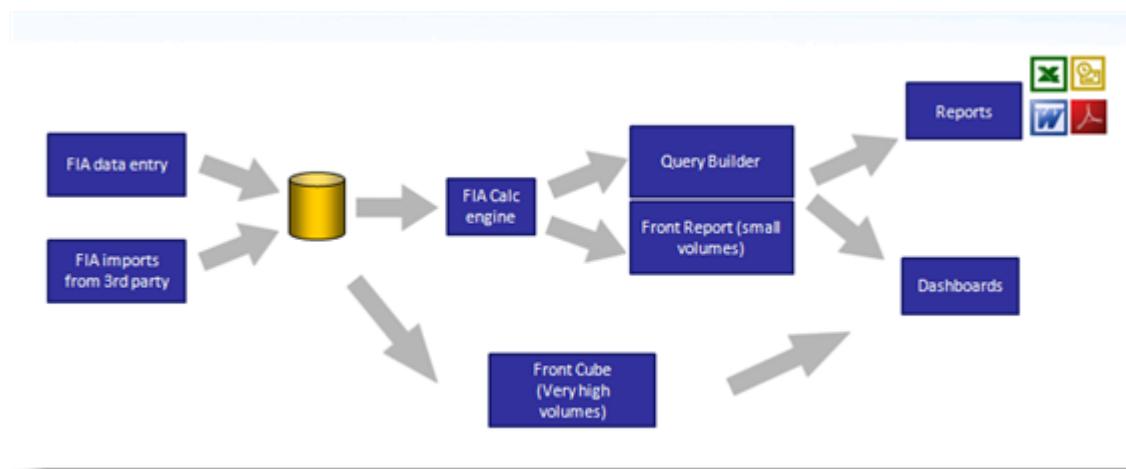
All other tasks are accomplished by procedures that operate on the table as a whole. Typical tasks include printing or performing statistical analysis, and may just require the user or programmer to identify the data set. Procedures are not restricted to only one behavior and thus allow extensive customization, which is controlled by statements and options defined within the procedures.

**eFront Script** offers macro programming extensions, which can be used to generate **eFront Script** code. Macro code in an **eFront Script** program undergoes **pre-processing**.

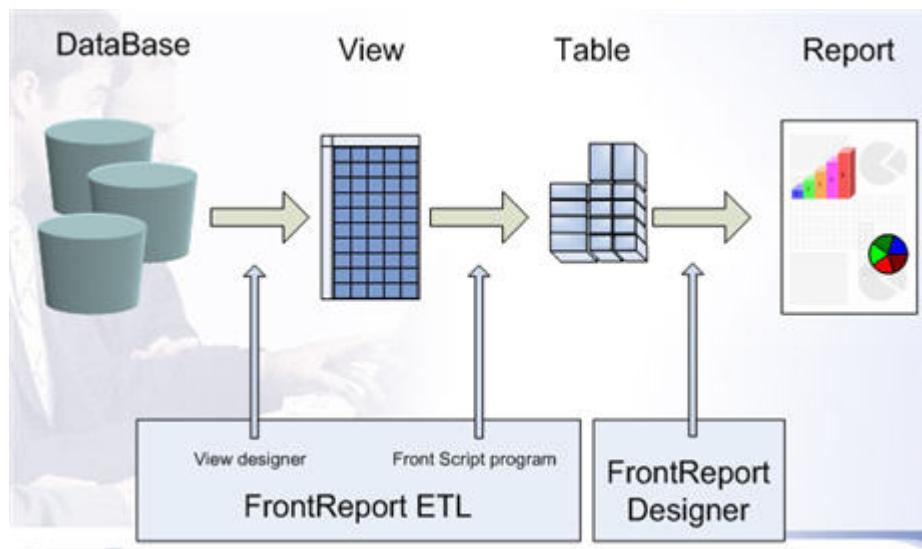
At runtime, procedure steps are compiled, interpreted and run in the sequence they appear in the **eFront Script** program. An **eFront Script** program requires the **eFront Report** environment to run.

## 2.2. eFront - Reporting solutions and eFront Report

**eFront** provides a set of decision-making and reporting solutions specifically designed for **financial businesses**. The entire reporting chain is automated: from your external databases, **EXCEL** files, or existing **eFront** databases, through to final production in the form of reports in different formats (WORD, EXCEL, PDF, HTML).



The process of creating reports using eFront Report is as follows:



**eFront Report** offers the Banking, Insurance and Private Equity sectors the opportunity to improve the production process of reports aimed at stakeholders (investors, regulators, customers, etc.), particularly by:

- enabling massive productivity gains in reporting tasks,
- accelerating the update frequency,
- enriching the details of information being transmitted,
- providing increased reliability of reported information.

**eFront Report** achieves report production **efficiency** and **quality** through the following work principles:

- Reduce time spent on **data extraction** from external databases or **eFront** databases: apply asynchronous batch processing (i.e. at night).
- Optimize and reduce time spent on **data processing**: create an **eFront Report** server-based datamart that bundles and quickly processes only the relevant report data.
- Guarantee **data freshness**: use cascading programs to easily update the **eFront Report** server-based datamart, while exploiting the links to the external databases.
- Guarantee **data security**: define data visibility based on user access rights and data regions.
- Reduce time spent on exchanges between client and server: create and edit **WORD** and **EXCEL** reports using the **eFront Report** server-based editing tools instead of client-based applications.
- Increase the quality of **WORD** and **EXCEL** reports: access optimized **VBA** and **EXCEL** libraries using the **eFront Report** server-based report editing tools.
- Reduce the **complexity of report authoring**: use the drag & drop based user interface. No need to dig into data extraction programs, SQL queries, data processing, programming formulas or complex spreadsheet operations.
- Achieve maximum ability to customize data extraction, data processing and report generation: use **eFront Script**, the **eFront Report** programming language.

## 2.3. Example eFront Script code

**eFront Script** uses DATA steps to read data, and procedures to analyze data and issue reports.

By default, a DATA step iterates through each record in a data table:

```
DATA USERS_1964;  
  
SET USERS (WHEREYEAR(BIRTHDAY) >= 1964);  
  
RUN;
```

The above DATA step creates a new data table **USERS\_1964** that includes those records from the data table **USERS** where **BIRTHDAY >= 1964**.

Amongst the procedures available, the PROC SORT step allows to sort data tables:

```
PROC SORT DATA = USERS_1964;  
  
BY LASTNAME;  
  
RUN;
```

The procedure below sorts the data table **USERS** by the **LASTNAME**. The following is an example of a complete program:

```
//Create the alias that points the existing table location  
LIBNAME TEST "\\\\EFRONT_SUP.2(Privé)\\TEST\\EXAMPLES\\TablesFormation";  
  
//Define the output table  
DATA TEST.USERS_AND_CONTRACTS;  
  
//Define the input tables  
MERGE TEST.USERS (IN=is_customer) TEST.CONTRACTS (IN=has_contract);  
  
//Define the common variable  
BY USER_ID;  
IF is_customer AND has_contract  
THEN OUTPUT;  
END;  
RUN;  
  
//Display the result  
PROC PRINT DATA = TEST.USERS_AND_CONTRACTS;  
RUN;
```

## 2.4. Type of data being manipulated

**eFront Script** manipulates data stored in column cells that are part of data tables. Columns can be referred to as variables. **eFront Script** handles a wide range of [variable types](#).

By default, all numeric variables are stored as real. However, it is possible to reduce precision. Date and datetime variables are numeric variables that inherit the C tradition and are stored as either the number of days (for date variables) or seconds (for datetime variables), starting from 1960-01-01 00:00:00.

## 2.5. Interdependency between steps and statements

The following table shows the interdependency between **steps** and **statements**, the main eFront Script programming objects:

Step	Valid statements	
DATA	ABORT statement BY statement COLUMN statement CONTINUE statement INFILE statement INLINE statement MERGE statement OUTPUT statement	PUT statement RECURSE statement RETURN statement SET statement STOP statement SQL statement TRACE statement
ODS		
PROC CONVERTCURRE	VAR statement	
PROC EFRONTACCRAULS		
PROC EFRONTIMPORT		
PROC EFRONTTABLES		
PROC EXCEL		
PROC EXECSQL		
PROC EXPORT		
PROC FORMAT	PICTURE statement	VALUE statement
PROC GCHART	DONUT statement DONUT3D statement HBAR statement HBAR3D statement	PIE statement PIE3D statement VBAR statement VBAR3D statement
PROC GPLOT	PLOT statement	
PROC IMPORT	COLUMN statement INFILE statement	

PROC MEANS	BY statement CLASS statement FIRST statement IRR statement JOIN statement LAST statement MAX statement	MEAN statement MIN statement MULTIPLE statement N statement SUM statement VAR statement NOBS statement
PROC MEMORY		
PROC OFFICE	SET statement	TEMPLATE statement
PROC PRINT	BY statement	SUM statement
PROC PRINTCOL	EXPORT statement	TITLE statement
PROC PRINTFORM	FORMAT statement	VAR statement
PROC SORT	BY statement	
PROC_SQLIMPORT		
PROC_SQLTABLE		
PROC TABULATE	BOX statement CLASS statement FORMAT statement	TABLE statement TITLE statement VAR statement
PROC TRANSPOSE	VAR statement	
PROC TRANSPOSE2	VAR statement	
STYLESHEET		
TABLE		

### 3. eFront Script syntax and notational conventions

Find details about:

- Notation of syntax elements
- eFront Script program grammar & punctuation
- Standard eFront Script data formats
- eFront Script naming rules

### 3.1. Notation of syntax elements

The syntax notation follows the below conventions:

- Placeholder variable names: output\_table
- eFront Script keywords: DATA
- Valid options and valid statements: </ table\_options>
- Placeholders for expressions: <expression>
- Alternative options: |

Here is an example:

```
DATAoutput_table (</table_options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

### 3.2. eFront Script program grammar & punctuation

- Case-insensitive.
- Line breaks are not meaningful.
- Statements are free-format: they can begin and end anywhere.
- Each statement must end in a semi-colon, e.g. RUN;
- 1-line comments start with a double slash: //**Comment**
- n-line comments start with a slash and star, and end with a star and slash, e.g. / \***Comment\*/**
- Values are enclosed by double or single quotations marks, e.g. "**Peter**".
- Valid **eFront Script** tokens must be used.

### 3.3. Standard eFront Script data formats

Format	Description
Real and integers	123, 12E+12, 123.12
Time	"18:30"t   "6:30PM"t   6:30:00"t
Date	"12JUN2005"d   "12062005"d

#### 3.3.1. Notes

The standard data format in eFront Cube programs differs from that used in eFront Script. The day, month and year must be separated with a dot (.), and the "d" identifier is not required. For example: "12.06.2005", or "12.JUN.2005".

## 3.4. eFront Script naming rules

Specific naming rules apply to:

- [Naming columns](#)
- [Naming formats](#)
- [Naming libraries](#)
- [Naming eFront Cube cubes](#)
- [Naming pictures](#)
- [Naming tables](#)
- [Naming transposed variables](#)
- [Referencing files](#)
- [Specifying paths](#)
- [Specifying pictures](#)

### 3.4.1. Naming columns

Column names must meet the following requirements:

- can contain only numbers, plain letters or underscores (no accents).
- cannot start with a number, except if the name is enclosed within square brackets.
- cannot contain any blank spaces within the name, except if the name is enclosed within square brackets.
- max length: 255 char.

### 3.4.2. Naming formats

When specifying format names, apply the following rules:

- Max name length : 32 char.
- Authorized symbols : character, number, underscore.
- Numbers are prohibited as a first or last character.
- The last character is a period ":".
- If the format applies to character strings, the first character of the format name is "\$".

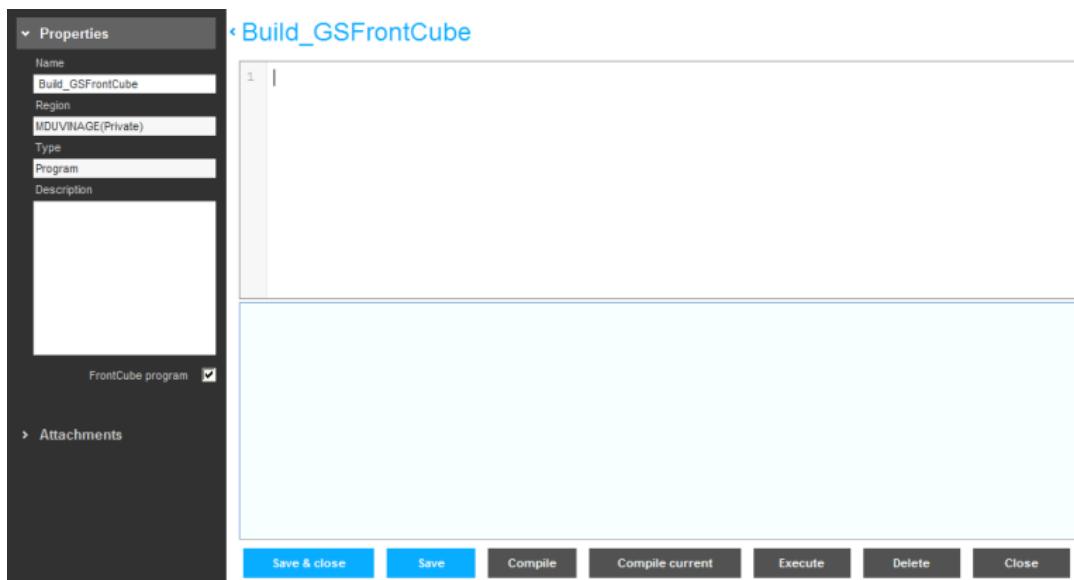
### 3.4.3. Naming libraries

When creating a library alias, apply the table name naming rules, and be careful not to use the names below as they are reserved **eFront Report** libraries:

- WORK: The WORK library points to a temporary folder, which is created by **eFront Report** at the beginning of a working session. The WORK library contains mainly temporary data sets and some utility files. By default, the entire WORK library is deleted at the end of each **eFront Script** job or session. The WORK library is user specific. If two users are using the WORK library at the same time, **eFront Report** creates two different WORK libraries.
- PUBLIC: refers to the public region, a folder that contains data shared by all users.
- PRIVATE: refers to the private region, a folder that contains data owned by the current user.
- SHARED: refers to the shared region, a folder that contains standard programs and standard tables shared by all users.
- INFO: refers to the Information region.
- DEFAULT: refers to the default region.
- CURRENT: refers to the current directory.
- DCE: refers to the output produced by the eFront Invest calculation engine for direct investments.
- FCE: refers to the output produced by the eFront Invest calculation engine for fund investments.
- CCE: refers to the output produced by the eFront Invest consolidated calculation engine for fund investments.

### 3.4.4. Naming eFront Cube cubes

eFront Cube programs are eFront Script programs whose **FrontCube program** feature is selected, as in the below example:



When creating eFront Cube programs, you need to use the following reserved names to make the cubes recognizable by the system:

- WORK(cube\_name)
- Session(cube\_name)
- Global(cube\_name)
- Universal(cube\_name)

These reserved cube names refer to **Cube levels**.

**Cube levels** categorize cubes, and serve as standard criteria that govern the use of the system memory, disk storage, cache, and the mechanism that clears cubes from the cache.

⚠ You cannot modify these rules.

**According to cube levels, the clearing of the cache is triggered:**

- when scripted actions explicitly specify it,
- when cubes are not used for a given period of time,
- when a defined memory limit is reached.

Depending on its level, a cube that is removed from the cache can then be saved on disk.

UNIVERSAL CUBE	
What is it?	persistent cube, which is shared among all the accounts.  Can be viewed by accessing the eFront Cube server's statistics page ( <a href="http://mycubeserver/qstats.aspx">http://mycubeserver/qstats.aspx</a> )
Where is it saved?	both on disk and in cache with no expiration.
When is it cleared from the cache?	- upon a scripted action explicitly triggered by an eFront Cube program, - when the defined memory limit is reached.

GLOBAL CUBE	
What is it?	persistent cube, which is shared among all the users of an account.  Can be viewed by accessing the eFront Cube server's statistics page ( <a href="http://mycubeserver/qstats.aspx">http://mycubeserver/qstats.aspx</a> )
Where is it saved?	both on disk and in cache with no expiration.
When is it cleared from the cache?	- upon a scripted action explicitly triggered by an eFront Cube program, - when the defined memory limit is reached.
SESSION CUBE	
What is it?	temporary cube, which exists during an end-user session or, if it is linked to a dashboard, during the time the dashboard is opened till the moment when it is closed.
Where is it saved?	in memory.
When is it cleared from the cache?	- at the end of the end-user's work session, - upon a scripted action explicitly triggered by an eFront Cube program,

WORK CUBE	
What is it?	temporary cube, which exists during a program's runtime.
Where is it saved?	in memory. Work cubes are never cached.
When is it cleared from the memory?	- at the end of the end-user's work session, - at the end of an eFront Cube program.

### 3.4.5. Naming pictures

When specifying picture names, apply the following rules:

- Max name length: 32 char
- Authorized signs: character, number, underscore.
- Numbers are prohibited as a first or last character.

### 3.4.6. Naming tables

Table names must satisfy the following requirements:

- only numbers, plain letters or underscores (no accents),
- no number at the first position,
- no blank spaces within the name,
- max length: 255 char,
- no **eFront Script** reserved name: WORK, PUBLIC, PRIVATE, SHARED, INFO, DEFAULT

Table names can be:

Table name	Explanation
tableName	If no library is specified, eFront Script creates (retrieves) the table tableName in the WORK library. The WORK library is a temporary folder that exists only during the eFront Script session.
libName.tableName	libName specifies the alias of the folder where eFront Script stores the table tableName
path\tableName	path specifies the access to the table tableName

### 3.4.7. Naming transposed variables

By default, the names of the variables being transposed are "COL1" "COL2", ..., "COLn". To change the default name of the transposed variables, use the PREFIX= option and the ID statement of the PROC TRANSPOSE step:

- Make a name by using the values of a variable: use the ID statement.
- Make a name by using prefixed numbers from 1 to n, the prefix being the name of the variable being transposed: use PREFIX=col.
- Make a name by using prefixed values of a variable, the prefix being the name of the variable being transposed: use PREFIX=col together with the ID statement.
- Make variable names out of numeric values: use the ID statement with an ID variable being a numeric variable. However, **eFront Script** variable names cannot begin with

a number. Thus, when the first character of the formatted value is numeric, the procedure prefixes an underscore to the value, truncating the last character of a 32-character value. Any remaining invalid characters are replaced by underscores. The procedure truncates to 32 characters any ID value that is longer than 32 characters when it uses that value to name a transposed variable. Signs, such as '+', '-', ';' are changed into an underscore.

- Assign a label to the name of the variable being transposed: use any of the above options to build the variable name together with the IDLABEL statement. The IDLABEL statement specifies the variables whose formatted values are to be used as labels. By default the label of the transposed variables is the formatted value.

Missing values:

If you use an ID variable that contains a missing value, then PROC TRANSPOSE writes an error message to the log. The procedure does not transpose observations that have a missing value for the ID variable.

### 3.4.8. Referencing files

In some circumstances, you might want to reference files being attached to documents, or files being located in eFront folders. To do so:

- Prefix the file name using "&". For example, "&Test.doc".
- Specify the path in the following way: \\REGION\...\test.xls

### 3.4.9. Specifying paths

Type of path	Usage	Example
Absolute path	\Region\folder-1\...\folder-n	%INCLUDE "\EFRONT_SUP.2(Private) \TEST\P_USERS_TO_INCLUDE";
Path relative to the current directory	folder-n	%INCLUDE "P_USERS_TO_INCLUDE"; (current directory=TEST)
Path relative to the program	%<library>% \folder-1\...\folder-n	

### 3.4.10. Specifying pictures

When specifying pictures, apply the following rules:

- Enclose the picture in single or double quotation marks
- Max length: 40 characters
- Use digit selectors, separator characters, and message characters
- Digit selectors define positions for significant digits. Use the **zero** as digit selector. If the picture exceeds the number of significant digits, leading zeros fill the vacant positions.
- If the number exceeds the picture, the number is not truncated!
- As separator characters, use the comma or the decimal point. The decimal point separates the fractional part of a number. The comma specifies the three-digit separator for a number.
- Message characters are non-numeric characters that print as specified in the picture.

## 4. eFront Script language elements

Each **language element topic** follows the following scheme:

- **Title:** Syntax element name
- **Description:** Description of the syntax element
- **Syntax:** Syntax of the element
- **Options:** Valid options used with the syntax element
- **Subordinate statements:** Subordinate statements used with the syntax element
- **Comment:** Additional information about the syntax element
- **Output:** Output being produced
- **Error messages:** Error messages linked to the invalid use of the syntax element
- **Related examples:** Links to related examples
- **See Also:** Links to related syntax elements
- **Conceptual information:** Links to conceptual information.

Language elements are:

- [Steps](#)
- [Embeddable steps](#)
- [Global statements](#)
- [Macro statements](#)
- [Step statements](#)
- [Options](#)
- [Macro controls](#)
- [Step controls](#)
- [Controls](#)
- [Functions - Operators - Values](#)
- [Style attributes and values](#)
- [Flags](#)
- [Variables](#)
- [XML reports](#)

## 4.1. Steps

Steps are independent program **blocks**, that start with an opening statement, and end with a closing statement. Each step is composed of a series of **statements**, that allow, according to the step's overall functionality, to specify the actions to undertake. Here is an example:

```
//opening statement
DATAoutput_table (</ table_options>);

//subordinate statement
SETinput_table (WHERE(<condition>));

//closing statement
RUN;
```

Steps cannot be nested. Here is the list of available steps:

Step	Main purpose
DATA	Creates a new data table. You can create new data tables from external files or from existing eFront Report tables. You can include all the data from the original table(s) or only part of it. You can also build entirely new tables without using pre-existing information.
ODS	Distributes output produced by the different programming steps to different destinations, such as MICROSOFT OFFICE applications, HTML pages or Listings.  The ODS step is the only step that includes other steps.
PROC CONVERTCURR	Converts amounts from input cube columns to output target cube columns using specified FX rates.  Applies only to eFront Cube programs.
PROC DELETE	Remove eFront Cube cube from eFront Cube server.  Applies only to eFront Cube programs.
PROC DIRECTCASHFLOWS	Generates a table via the Calculation Engine that returns either all the direct cash flows (transaction ID's) of the database or the cashflows for a selection of instruments and/or transactions.
PROC EFRONTACCRUALS	Generates a table that calculates accruals for a given date.

PROC EFRONTBLOB	Exports data from a FIA blob.
PROC EFRONTDASHBOARD	Exports a dashboard as a PDF file.
PROC EFRONTFOLDER	Extracts data related to a specific object in a data folder.
PROC EFRONTMAIL	Triggers the automatic sending of emails.
PROC EFRONTIMPORT	Imports data from an eFront Report data table.
PROC EFRONTPACKAGES	Retrieves information about the packages integrated into an eFront application and the package options.
PROC EFRONTIMPORT_AJX	Executes a specific sheet from the file of a given FIA import registered in the system
PROC EFRONTMENUS	Retrieves all the menus and sub-menus for the application users.
PROC_EFRONTPROFILES	Retrieves the user profiles defined for the application.
PROC EFRONTREGIONS	Retrieves information about user regions and region access rights
PROC EFRONTTABLES	Generates tables that describe the data model. These meta-tables can describe tables, columns, relations, and indexes.
PROC EFRONTUSERPROFILES	Retrieves the profiles of the users of the application
PROC ERROR	Stops the execution of a program by raising an exception with a custom message.
PROC EXPORT	Exports an eFront Report data table or view formatted as .CSV file to your local server disk. You can export the table structure only or the table structure including the data.
PROC EXPORTCHART	Extracts all the charts of an EXCEL file in JPEG format.
PROC EXPORTEXCEL	Exports one or more eFront Report data tables to one or more EXCEL sheets of the same EXCEL workbook (one table per sheet).
PROC EXCEL	Generates an .XLS file with data from an eFront Report input table.
PROC FALIBRARY	Imports an eFront Analytics data table / data cube into eFront Report.
PROC FAQUERY	Exports data a QueryBuilder query into an eFront Report table.
PROC FORMAT	Creates customized formats for data display.
PROC FUNDCASHFLOWS	Generates a table via the Calculation Engine that returns the ID's of all the investor operations, and investor operations by shares, if the fund manages shares.
PROC GCHART	Creates a chart from an eFront Report data table.
PROC GPLOT	Creates a plots & lines chart from an eFront Report data table. Charts can be made of pies, bars, donuts.
PROC MEANS	Calculates statistics for an eFront Report data table.
PROC MEMORY	Displays data table size, and unloads data tables from memory. tables.
PROC OFFICE	Generates an WORD, EXCEL or XML file based on an appropriate template and fed with data from one or more eFront Report data tables.
PROC PRINT	Prints an eFront Report data table, using all or some of the variables.

<a href="#">PROC PRINTCOL</a>	Writes a column of a data table into a multi-column print table.
<a href="#">PROC PRINTFORM</a>	Prints a form.
<a href="#">PROC SENDTOIC</a>	Sends documents (PDF, DOCX) from eFront Invest to Investment Café Classic.
<a href="#">PROC SETFILTER</a>	Filters automatically an eFront Cube cube according to the defined filterings expressions, at the moment when it is accessed by a dashboard.
<a href="#">PROC SORT</a>	Sorts tables by the values of one or more character or numeric variables.
<a href="#">PROC SQLEXEC</a>	Executes a SQL query on an external database.
<a href="#">PROC SQLIMPORT</a>	Imports directly SQL data into an eFront Report table.
<a href="#">PROC SQLTABLE</a>	Exports an eFront Report table to a SQL database.
<a href="#">PROC TABULATE</a>	Creates a pivot table, and calculates statistics.
<a href="#">PROC TRANSPOSE</a>	Creates an output table by restructuring the input table, transposing selected columns into lines (observations). Useful procedure when preparing reports.
<a href="#">PROC TRANSPOSE2 step</a>	Creates an output table by restructuring the input table: the values of a selected column are used to create additional table columns, and the values of another selected column in the initial table are written into the new columns.
<a href="#">PROC UPDATE</a>	Merges two eFront Cube cubes (left join merge). The PROC UPDATE step is very similar to the DATA MERGE statement with the limitation that it is impossible to add or remove rows. In comparison with the DATA MERGE statement, the PROC UPDATE step has a much better performance as the impacted columns are not recalculated.
<a href="#">PROC UPDATEFOLDER</a>	Updates a WebEdge folder.
<a href="#">STYLESHEET</a>	Defines different styles applicable to programming step elements.
<a href="#">TABLE</a>	Dispatches printed output in an HTML table.

## 4.1.1. DATA step

### 4.1.1.1. Description

Independent program block. Always begins with the DATA statement and ends with a RUN statement. Creates a new data table. You can create new data tables from external files or from existing **eFront Report** tables. You can include all the data from the original table(s) or only part of it. You can also build entirely new tables without using existing information.

### 4.1.1.2. Syntax

```
DATAoutput_table (</ table_options>);

</ subordinate statements>

RUN;
```

### 4.1.1.3. Subordinate statements

Statement	Description
<a href="#">ABORT</a>	Stops the execution of the program.
<a href="#">COLUMN</a>	Adds a new column.
<a href="#">CONTINUE</a>	Exits the current control and jumps to the next statement.
<a href="#">INFILE</a>	Reads all the lines of an external file (.CSV or .TXT).
<a href="#">INLINE</a>	Inserts data lines in an existing table. Must be used together with the INFILE statement.
<a href="#">MERGE...BY</a>	Reads simultaneously several input tables (junction of tables).
<a href="#">OUTPUT</a>	Reads all the lines of 1 or more existing tables into the output table.
<a href="#">PUT</a>	Writes data values to a special buffer from which they can be written to the data component.
<a href="#">RECURSE</a>	Reads recursively lines from 1 or more input tables, according to Parent/Child relations.
<a href="#">RETURN</a>	
<a href="#">SET</a>	Reads all the lines of 1 or more existing tables into the output table (concatenation of tables).
<a href="#">STOP</a>	Stops execution of the current step, starts execution of the next step.

### 4.1.1.4. Comment

- `output_table` : refer to [Table names](#).
- If `output_table` already exists, the new table replaces the existing one; **eFront Script** issues no warning.
- By default, **eFront Script** adds a first column to each data table: the OBS column (OBS: observation). OBS values indicate the line position within the `output_table`.
- To display the table on screen, use the [PROC PRINT](#) step.
- To store the table permanently, use the [LIBNAME](#) statement.
- Use [controls](#) to process groups of statements.
- You can only specify one single `output_table`.
- SET | MERGE | INFILE | RECURSE | OUTPUT (mutually exclusive).

- To create a table from scratch, use the **COLUMN** statement.

#### 4.1.1.5. Output

No direct output.

#### 4.1.1.6. Error messages

Message	Explanation
ERROR(0113): Table name expected	The table name has been left out in the DATA statement, or erroneously assigned using an "=" sign.
0001 DATA_NONE	No data has been assigned to table
ERROR(0111): ';' expected	';' has been left out

#### 4.1.2. ODS step

##### 4.1.2.1. Description

Independent program block. Always begins with the ODS statement and ends with and ODS END statement. Distributes the output of the other programming steps to different destinations, such as **MICROSOFT OFFICE** applications, **HTML** pages or **Listings**.

##### 4.1.2.2. Syntax

ODSdestination </ option>;

</ subordinate steps>;

ODSdestinationEND;

##### 4.1.2.3. Comment

- destination : OFFICE | HTML | LISTING
- <subordinate steps>: The ODS step must include the programming steps that produce the output to be sent to destination.
- The ODS step is the only step that includes other steps.

#### 4.1.2.4. Options

Option	Description
FILE=	<p>Relative or absolute path to file. Ex. :</p> <p>FILE= "OutputFile.xls"</p> <p>FILE="\Folder-1\...Folder-n\OutputFile.xls"</p> <p>The file extension must correspond to the destination.</p>

#### 4.1.2.5. Output

Generates an output file.

#### 4.1.2.6. Error messages

Message	Explanation
---------	-------------

### 4.1.3. PROC CONVERTCURR step

#### 4.1.3.1. Description

Independent program block. Always begins with the PROC CONVERTCURR statement and ends with a RUN statement. Converts amounts from input columns to target output columns using the specified FX rates.

 Applies only to eFront Cube programs.

#### 4.1.3.2. Syntax

PROC CONVERTCURR </ options>;

</ subordinate statements>;

RUN;

#### 4.1.3.3. Options

Option	Description
--------	-------------

DATA=input_cube	input_cube: Specifies the name of the cube which contains the columns to convert, typically transaction positions cube.
OUT=output_cube	output_cube: Specifies the name of the output cube which contains the converted columns.   Typically the same cube as the DATA cube.
FXRATES=cubeContainingTheFXRATES	cubeContainingTheFXRATES: Specifies the name of the cube which contains the already calculated FX rates.
CURRENCY_COLUMN="currency_column_name_in_DATA_cube":	"currency_column_name_in_DATA_cube": Specifies the name of the column in the DATA cube which refers to the currency of the amount to be converted.
FXRATES_CURRENCY_COLUMN="source_currency_column_in_FXRATES_cube"	"source_currency_column_in_FXRATES_cube": Specifies the name of the column of the FXRATES cube which contains the source currency.
FXRATES_RATE_COLUMN="FX_rate_column_name_in_FXRATES_cube":	"FX-rate-to-apply_column_name_in_FXRATES_cube": Specifies the name of the column of the FXRATES cube which contains the FX to apply.

#### 4.1.3.4. Subordinate Statements

Statement	Description
VAR	Identifies the input columns containing the amounts to be converted.

#### 4.1.3.5. Comment

- Using the PROC CONVERTCURR step is the most efficient way to apply FX rates to a large number of columns.

#### 4.1.3.6. Output

No printed output.

#### 4.1.3.7. Error messages

Message	Explanation
---------	-------------

### 4.1.4. PROC DELETE step

#### 4.1.4.1. Description

Independent program block. Always begins with the PROC DELETE statement and ends with a RUN statement. Deletes an eFront Cube cube from the eFront Cube server.



Applies only to eFront Cube programs.

#### 4.1.4.2. Syntax

```
PROC DELETE DATA=cube_name;
```

```
RUN;
```

#### 4.1.4.3. Output

Deleted eFront Cube cube.

#### 4.1.4.4. Error messages

Message	Explanation
---------	-------------

### 4.1.5. PROC DIRECTCASHFLOWS step

#### 4.1.5.1. Description

Independent program block. Always begins with the PROC DIRECTCASHFLOWS statement and ends with a RUN statement. Generates an eFront Report data table that returns all the direct cash flows (transaction IDs) for a collection of instrument IDs and/or transaction IDs. The data is directly extracted from the database via the **Calculation Engine (DCE)**.

PROC DIRECTCASHFLOWS returns a list of direct cash flows - information about positions is not included! The advantage of using the table produced by the eFront Script step lies in an increased performance in extracting information.

 If you need a direct cash flow table, which also contains the positions, we recommend you use the table [FCE.DIRECTCASHFLOWS](#). If you do so, you don't need to use the PROC DIRECTCASHFLOWS.

 For each instrument, the step generates a "ghost" transaction. This transaction, which is an adjustment, enables the calculation of instrument positions by adding up column content, even for values, which depend on FX variations such as the valuation.

 Not available for eFront Cube programs.

#### 4.1.5.2. Syntax

PROC DIRECTCASHFLOWS </ options>;

RUN;

#### 4.1.5.3. Options

Option	Description
TRANSACTIONS=transaction	optional; transaction: --> a collection of transaction ID's or --> a SQL query, such as 'SELECT IQID from...'
INSTRUMENTS=instrument	optional; instrument: --> a collection of instrument ID's or --> a SQL query, such as 'SELECT IQID from...'
OUT=lib.table	mandatory; output table

#### 4.1.5.4. Comment

The PROC DIRECTCASHFLOWS returns transactions according to the following priority:

1. Transactions for the specified instruments (INSTRUMENTS option).
2. Transactions corresponding to the SQL query expressed in the TRANSACTIONS option.
3. Transactions corresponding to the transactions' IDs transmitted in the TRANSACTIONS option.

 If no options (INSTRUMENTS, TRANSACTIONS) have been specified, PROC DIRECTCASHFLOWS returns all the transactions recorded in the database.

 The ID of the generated ghost transactions corresponds to the instrument ID.

 If values have been specified for both options (INSTRUMENTS, TRANSACTIONS), PROC DIRECTCASHFLOWS inquires the database with all the instrument ID's, but performs calculations only on the transactions specified in the TRANSACTIONS option.

#### 4.1.5.5. Applicable macros

You can modify the default behavior of PROC DIRECTCASHFLOWS using the following macros:

- If you define the %REPORT\_DATE macro, the date to be used is the value of the variable %DATE, otherwise the system uses the current date.
- If you define the %USE\_DRAFT, draft transactions are used by the calculation engine, otherwise they are ignored.
- If you define a value for the variable %CE\_CURRENCY , this value will be used as currency (ex.: 'EUR', 'USD').
- If you define the %USE\_INVESTEE\_CURR, calculations are done in investee currency, otherwise calculations are done in investor currency.
- If you define the %CALCULATE SHARES macro, the ghost transactions integrate the number of diluted investment instruments.

#### 4.1.5.6. Output

Generates an eFront Report data table.

#### 4.1.5.7. Error messages

Message	Explanation

#### 4.1.6. PROC EFRONTACCRUALS step

##### 4.1.6.1. Description

Independent program block. Always begins with the PROC EFRONTACCRUALS statement and ends with a RUN statement. Generates a table that calculates **accrued interests** per instrument for a given date. The generated table includes 8 columns:

- COLUMN "LOANACCRUALS\_PRINCIPAL" LABEL="Principal{F}Nominal" TYPE="FLOAT";
- COLUMN "LOANACCRUALS\_WRITEOFF" LABEL="Writeoffs{F}Writeoffs" TYPE="FLOAT";
- COLUMN "LOANACCRUALS\_PIK" LABEL="PIK{F}Intérêts capitalisés" TYPE="FLOAT";
- COLUMN "LOANACCRUALS\_BALANCE" LABEL="Balance{F}Balance" TYPE="FLOAT";
- COLUMN "LOANACCRUALS\_DUEINTEREST" LABEL="Due interests{F}Intérêts dûs" TYPE="FLOAT";
- COLUMN "LOANACCRUALS\_PAIDINTEREST" LABEL="Paid interests{F}Intérêts payés" TYPE="FLOAT";
- COLUMN "LOANACCRUALS\_CASHACCRUAL" LABEL="Cash accruals{F}Coupons courus cash" TYPE="FLOAT";
- COLUMN "LOANACCRUALS\_PIKACCRUAL" LABEL="PIK accruals{F}Coupons courus PIK" TYPE="FLOAT";

Note that you can generate accruals per instrument investment round.

 Starting from eFront Invest 7.1, the columns have a different name: you need to add "2" at the end to know that the amount is in instrument currency. Example:

- COLUMN "LOANACCRUALS\_PRINCIPAL2" LABEL="Principal{F}Nominal" TYPE="FLOAT";

- COLUMN "LOANACCRUALS\_WRITEOFF2" LABEL="Writeoffs{F}Writeoffs"  
TYPE="FLOAT";

#### 4.1.6.2. Syntax

PROC EFRONTACCRUALS (</ **table options**> </ **options**>);

RUN;

#### 4.1.6.3. Options

Option	Description
DATE =date	Identifies the date for which accrued interests are calculated; mandatory option
ENDOFDAY	
ID = col_instrument	Identifies the column where to get the id_instrument.
ROUNDDID = col	Identifies the investment round for the instrument.
MISSING	
OUT = output_table	Identifies the output_table. If you don't specify an output_table, eFront Script overwrites the table_to_process
TYPE = event_type	Filters loan events by type.  Type can either be a string containing one or several event types (separated by a semicolon), or it can be a collection of event types.
USEDRAFT	Use draft transactions.
USERECURRING	Use recurring transactions.

#### 4.1.6.4. Output

HTML table.

#### 4.1.6.5. Error messages

Message	Explanation

## 4.1.7. PROC EFRONTBLOB step

### 4.1.7.1. Description

Independent program block. Always begins with the PROC EFRONTBLOB statement and ends with a RUN statement. Exports data from a FIA blob field or a collection of FIA blob fields into an **eFront Report** table.

### 4.1.7.2. Syntax

```
PROC EFRONTBLOB </ options>;  
</ subordinate statements>;  
RUN;
```



The options used with the keyword EFRONTBLOB will vary depending on whether you want to export individual blob fields or a table containing blob fields.

### 4.1.7.3. Options for exporting groups BLOB fields

Option	Description
DATA="table_containing_blob_fields"	"table_containing_blob_fields":  Refers to the table that contains the blob fields. Usually you would extract this table through an SQL request, store the results in a macro variable, and then use the macro variable in EFRONTBLOB step.  Mandatory option;
ID="blob_field_column"	Refers to the column within the blob fields' table, which contains the blob field values.  Mandatory option;

CLASS="blob_field_class"	<p>"blob_field_class":</p> <p>Refers to the class linked to the blob fields.</p> <p>When creating blob fields in eFront Invest, you associate them with classes, which allows you to regroup them into blob field groups. The 'CLASS' parameter refers to this class.</p> <p>Mandatory option;</p> <p>Here are some examples of classes:</p> <ul style="list-style-type: none"> <li>Company --&gt; Account</li> <li>Deal --&gt; Project</li> <li>Portfolio --&gt; Portfolio</li> <li>Fund --&gt; Fund</li> <li>Appointment --&gt; Appointment</li> </ul>
PATH="section_name"	<p>"section_name":</p> <p>Refers to an expression containing the section name of the data in the blob definition (section used when you created the data).</p> <p>Mandatory option;</p>
PREFIX="prefix_for_output_table"	<p>"prefix_for_output_table":</p> <p>Refers to the prefix to be used in the eFront Report output table to signal the blob fields.</p> <p>Optional option;</p>
OUT = tableout	<p>tableout:</p> <p>Refers to the eFront Report output table.</p> <p>Mandatory option</p>

#### 4.1.7.4. Options for exporting individual BLOB fields

Option	Description
BLOB="blob_value"	<p>"blob_value":</p> <p>Refers to an expression that contains the blob value. Usually you extract it from a column, store it in a macro and you use this macro in the blob expression.</p> <p>Mandatory option</p>

FORMAT="blob_field_format"	"blob_field_format":  Refers to the format to be applied to the values of the blob field in the eFront Report output table.  Optional option;
PATH="section_name"	"section_name":  Refers to an expression containing the section name of the data in the blob definition (section used when you created the data).  Mandatory option
OUT = tableout	tableout:  Refers to the eFront Report output table.  Mandatory option

#### 4.1.7.5. Subordinate statements

Option	Description
VAR	

statement

#### 4.1.7.6. Comment

- Using the PROC EFRONTBLOB step is the most appropriate way to export data from a blob field.

#### 4.1.7.7. Output

An output file.

#### 4.1.7.8. Error messages

Message	Explanation

## 4.1.8. PROC EFRONTDASHBOARD step

### 4.1.8.1. Description

Independent program block. Always begins with the PROC EFRONTDASHBOARD statement and ends with a RUN statement. Exports a dashboard as a PDF file.



Not available for eFront Cube programs.

### 4.1.8.2. Syntax

```
PROC EFRONTDASHBOARD DASHBOARD =
dashboard_path(dashboard_parameters) PAGES =list_of_pages;
```

```
</ subordinate statements>;
```

```
RUN;
```

### 4.1.8.3. Options

Option	Description
DASHBOARD="dashboard_path"	<p>"dashboard_path":</p> <p>The path to the dashboard that is to be exported as a PDF file.</p> <p>Mandatory;</p>
PAGES="list_of_pages"	<p>"list_of_pages":</p> <p>A list of dashboard sheets that will be exported. Available formats:</p> <ul style="list-style-type: none"> <li>• sequence: e.g. 1,2,3,5</li> <li>• range: e.g. 1-5</li> <li>• sequence and range: e.g. 1,3,5-7</li> </ul> <p>Optional;</p>
dashboard_parameters	<p>The parameters used to filter the dashboard's data.</p> <p>The parameters must be placed between parentheses (), immediately after the dashboard path. The format is as follows:</p> <pre>(Parameter_name1='parameter_value1' Parameter_name2='parameter_value2')</pre>

#### 4.1.8.4. Comment

- The PAGES option is not necessary if all dashboard pages need to be exported.

#### 4.1.8.5. Output

PDF file of the specified dashboard.

#### 4.1.8.6. Error messages

Message	Explanation
---------	-------------

### 4.1.9. PROC EFRONTFOLDER step

#### 4.1.9.1. Description

Independent program block. Always begins with the PROC EFRONTFOLDER statement and ends with a RUN statement. Extracts data related to a specific object in a data folder.

#### 4.1.9.2. Syntax

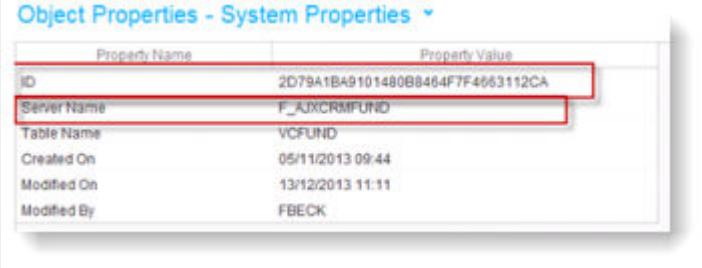
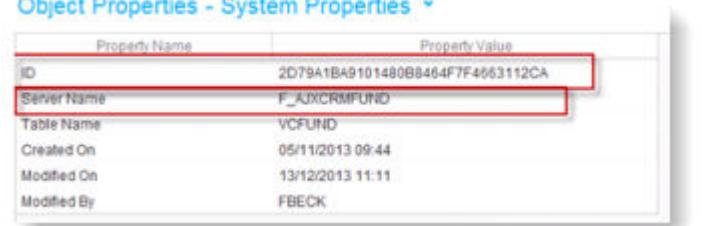
```
PROC EFRONTFOLDER FOLDER = folder_nameID =folder_idPROPERTIES  
=folder_properties;
```

```
</ subordinate statements>;
```

```
RUN;
```

#### 4.1.9.3. Options

Option	Description
--------	-------------

FOLDER="folder_name"	<p>"folder_name":</p> <p>Refers to the folder name of the data folder, from which to extract data.</p>  <p><i>i</i> To identify the generated tables, use the snippet below:</p> <pre>%FOR %T %IN (%folder_name) TRACE %T; %NEXT;</pre> <p>Mandatory option;</p>
ID="folder_id"	<p>"folder_id":</p> <p>Refers to the unique identification number of the folder from which to extract the data.</p>  <p>Mandatory option;</p>
PROPERTIES="folder_properties"	<p>"folder_properties":</p> <p>Refers to a property to be used to filter folder data. The folder properties differ from folder to folder.</p> <p>Mandatory option;</p>

#### 4.1.9.4. Comment

- You can use the EFRONTFOLDER step to extract cashflows, for example.

#### 4.1.9.5. Output

Data tables of the specified folder.

#### 4.1.9.6. Error messages

Message	Explanation
---------	-------------

### 4.1.10. PROC EFRONTIMPORT step

#### 4.1.10.1. Description

Independent program block. Always begins with the PROC EFRONTIMPORT statement and ends with a RUN statement. Imports data from an eFront Report data table.

 Not available for eFront Cube programs.

#### 4.1.10.2. Pre-requisite

An [import mask has been created](#) that can be referenced from within the PROC EFRONTIMPORT step.

#### 4.1.10.3. Syntax

PROC EFRONTIMPORT DATA = table\_to\_import (</ table\_options> </ options>);

RUN;

#### 4.1.10.4. Options

Option	Description
NAME=""	Name of the import mask.  Example:  Name = "##BLOOMBERG##"
DETAILS	Optional; Creates the %OUTPUTLOG macro with the import log.

#### 4.1.10.5. Output

No direct output.

#### 4.1.10.6. Error messages

Message	Explanation
ERROR: Syntax error : invalid PROC qualifier	

### 4.1.11. PROC EFRONTIMPORT\_AJX step

#### 4.1.11.1. Description

Independent program block. Always begins with the PROC EFRONTIMPORT\_AJX statement and ends with a RUN statement. Executes a specific sheet from the file of a given FIA import registered in the system. Imports an eFront Report table in a specific sheet from the file of a given FIA import registered in the system.



Not available for eFront Cube programs.

#### 4.1.11.2. Prerequisite

The eFront Invest import, which is referenced from within the PROC EFRONTIMPORT\_AJX step, exists. The FrontReport table table\_to\_import must exists.

#### 4.1.11.3. Syntax

PROC EFRONTIMPORT\_AJX DATA = table\_to\_import (</ table\_options> </ options>);

RUN;

#### 4.1.11.4. Options

Option	Description
NAME=""	<p>Name of the FIA import.</p> <p>Example:</p> <p>Name = " "</p>

SHEETNAME=""	Name of the EXCEL sheet referred to, which is part of the FIA import  Example:  Sheetname = " "
DETAILS	Optional; Creates the %OUTPUTLOG macro with the import log.

#### 4.1.11.5. Comment



It is important that the eFront Report data table into which you import the data from the EXCEL spreadsheet has the same DATA column information as the spreadsheet (without the ef\$class and ef\$subclass information).

A	B	C	D	E	F	G	H	I	J
ef\$class	ef\$subclass	ef\$key	ef\$col	ef\$col	ef\$key	ef\$key	ef\$key	ef\$key	ef\$col
FundOperation	WithShares	Fund.Fund	Investors (BP).Subscriber.Fund	Investment	Investors (BP).Share Mgmt Fees In Commitment	Date	Operation	Index	Type
									Investors (BP).Fund share.Category

↓

FUND_NAME	INVESTOR_NAME	MGT_FEES	OPE_DATE	OPERATION	INDEX	TYPE	SHARE
INFRA II	01 European	463047.864205954	05/12/2011	Call 19	1	Call	01 UNIT

#### 4.1.11.6. Output

EXCEL sheet in the standard FIA import.

#### 4.1.11.7. Error messages

Message	Explanation
ERROR: Syntax error : invalid PROC qualifier	

### 4.1.12. PROC EFRONTMAIL step

#### 4.1.12.1. Description

Independent program block. Always begins with the PROC EFRONTMAIL statement and ends with a RUN statement. Triggers the automatic sending of emails. The system generates one mail per line in the table. You can define the columns, which are associated with the FROM, TO, CC, BBC, etc. of the mails.

## 4.1.12.2. Prerequisite

- In order to be able to send emails, the smtp information in the **Config** file needs to be properly set up.

The easiest way to do this would be to use FrontAdmin in eFront business solution's installation directory (e.g. C:\Efront\website\deployment). Smtp information is located in Section 6 of FrontAdmin's Advanced Settings.

## 4.1.12.3. Syntax

PROC EFRONTMAIL DATA = table\_name (</table\_options></ options>);

RUN;

## 4.1.12.4. Options

Data filter	Description
WHERE(<condition>)	Filters the data to be kept. To write the <condition>, use available <a href="#">Functions - Operators - Values</a> .
FROM=<column_name>	Selects the column, which contains the sender
TO=<column_name>	Selects the column, which contains the receiver. Use the ';' separator to enter multiple values.
CC=<column_name>	Selects the column, which contains the receiver of the carbon copy (CC). Use the ';' separator to enter multiple values.
BCC=<column_name>	Selects the column, which contains the receiver of the blind carbon copy (BCC). Use the ';' separator to enter multiple values.
SUBJECT=<column_name>	Selects the column, which contains the subject line.
BODY=<column_name>	Selects the column, which contains the email message in Plain Text format.
HTMLBODY=<column_name>	Selects the column, which contains the email message formatted in HTML.
ATTACHMENTS=<column_name>	Selects the column, which contains the full paths to the files to be sent as attachments. You need to separate multiple attachments (full path references) by semicolons.
DETAILS	Optional; Creates the %OUTPUTLOG macro with the import log.

#### 4.1.12.5. Output

Automatically sent emails.

#### 4.1.12.6. Error messages

Message	Explanation
---------	-------------

### 4.1.13. PROC EFRONTMENUS step

#### 4.1.13.1. Description

Independent program block. Always begins with the PROC EFRONTMENUS statement and ends with a RUN statement. Retrieves all the menus and sub-menus for an application user.

 Not available for eFront Cube programs.

#### 4.1.13.2. Syntax

```
PROC EFRONTMENUSPROFILE=GETUSERPROFILE(GETUSERID());
```

```
RUN;
```

#### 4.1.13.3. Options

Option	Description
PROFILE=GETUSERPROFILE(GETUSERID())	GetUserProfile(GetUserID): retrieves the IQID of the user for whom to retrieve menus and sub-menus mandatory;
OUT=table	output table (ex.: WORK.MENUS) mandatory;

#### 4.1.13.4. Comment

- The results returned are based on the **Menu Administration** as available up from V9.0, and which configuration must have been **saved at least once** to produce results.

The screenshot shows the 'Menu administration' page with the following details:

**Menus**

Menu	Caption
NEW	New(F)Nouveau
OPEN	Open(F)Ouvrir
DASHBOARDS	Dashboards(F)Tableaux de bord
DIRECT	Companies(F)Sociétés
FOF	Funds(F)Fonds
CASHFLOW	Cashflow forecasting
UNQUOTED	Statistics(F)Statistiques
PM	Portfolio Management
GL	General Ledger(F)Comptabilité
COMPLIANCE	Compliance(F)Conformité

**Menu items**

Caption	Icon
Dashboard(F)Tableau de bord	images/skin/METRO_ajxrep-chart3.png
Component(F)Composant	images/ajxrep-studio3.gif
Hedge Instrument(F)Instrument de couverture	images/hedgeinstrument.png

Buttons at the bottom: Save & Close (disabled), Save.

#### 4.1.13.5. Output

eFront Report table.

#### 4.1.13.6. Error messages

Message	Explanation

### 4.1.14. PROC EFRONTPACKAGES step

#### 4.1.14.1. Description

Independent program block. Always begins with the PROC EFRONTPACKAGES statement and ends with a RUN statement. Retrieves information about the packages integrated into an eFront application and the package options.

Generates a table that includes 15 columns:

- COLUMN "LEVEL" LABEL="Level{F}Niveau" TYPE="STRING";
- COLUMN "PROFILE\_ID" LABEL="Profile ID{F}Id profil" TYPE="STRING";
- COLUMN "PROFILENAME" LABEL="Name{F}Nom" TYPE="STRING";
- COLUMN "USER\_ID" LABEL="User ID{F}Id utilisateur" TYPE="STRING";
- COLUMN "LOGIN" LABEL="User Login{F}Login utilisateur" TYPE="STRING";
- COLUMN "PACKAGENAME" LABEL="Package name{F}Nom package" TYPE="STRING";
- COLUMN "PACKAGEACCESS" LABEL="Access{F}Accès" TYPE="STRING";
- COLUMN "KEYNAME" LABEL="Key name{F}Nom clé" TYPE="STRING";
- COLUMN "KEYCAPTION" LABEL="Key caption{F}Label clé" TYPE="STRING";
- COLUMN "KEYVALUE" LABEL="Key value{F}Accès lecture" TYPE="STRING";
- COLUMN "NEW" LABEL="New{F}Nouveau" TYPE="BOOLEAN";
- COLUMN "MODIFY" LABEL="Modify{F}Modifier" TYPE="BOOLEAN";
- COLUMN "DELETE" LABEL="Delete{F}Supprimer" TYPE="BOOLEAN";
- COLUMN "DUPLICATE" LABEL="Duplicate{F}Dupliquer" TYPE="BOOLEAN";
- COLUMN "KEYDESCR" LABEL="Key description{F}Description clé" TYPE="STRING";

 Not available for eFront Cube programs.

#### 4.1.14.2. Syntax

PROC EFRONTPACKAGES </ options>;

RUN;

#### 4.1.14.3. Options

Option	Description
--------	-------------

LEVEL =level	level: refers to the level of package usage. Possible values are: "ACCOUNT": retrieves package options at account level "PROFILE": retrieves package options at profile level (if used together with a profile IQID, retrieves package options for a particular profile) "USER": retrieves package options at user level (if used together with a user IQID, retrieves package options for a particular user) optional;
PROFILE =profile_id	profile_id: IQID of a profile; Retrieves, if used together with LEVEL="PROFILE", the package options of a particular profile. optional;
PACKAGE =package	package: name of package; (example: PCKAJX_DASHBOARDS) Retrieves all the package options of the package. optional;
USER =user_id	user_id: IQID of a user; Retrieves, if used together with LEVEL="USER", the package options of a particular user. optional;
OUT=table	mandatory; output table (ex.: WORK.PACKAGES)

#### 4.1.14.4. Output

eFront Report table.

#### 4.1.14.5. Error messages

Message	Explanation

## 4.1.15. PROC EFRONTPROFILES step

### 4.1.15.1. Description

Independent program block. Always begins with the PROC EFRONTPROFILES statement and ends with a RUN statement. Retrieves the user profiles defined for the application.

Generates a table that includes 3 columns:

- COLUMN "PROFILE\_ID" LABEL="Profile ID{F}Id profil" TYPE="STRING";
- COLUMN "PROFILENAME" LABEL="Name{F}Nom" TYPE="STRING";
- COLUMN "ADMIN" LABEL="Administrator{F}Administrateur" TYPE="STRING";



Not available for eFront Cube programs.

### 4.1.15.2. Syntax

PROC EFRONTPROFILES </ options>;

RUN;

### 4.1.15.3. Options

Option	Description
PROFILE =profile_id	profile_id: IQID of the profile for which to turn up the information optional;
OUT=table	mandatory; output table (ex.: WORK.PROFILES)

### 4.1.15.4. Output

eFront Report table.

### 4.1.15.5. Error messages

Message	Explanation

## 4.1.16. PROC EFRONTREGIONS step

### 4.1.16.1. Description

Independent program block. Always begins with the PROC EFRONTREGIONS statement and ends with a RUN statement. Retrieves information about user regions and region access rights.

Generates a table that includes 9 columns:

- COLUMN "USER\_ID" LABEL="User ID{F}ID utilisateur" TYPE="STRING";
- COLUMN "LOGIN" LABEL="User Login{F}Login utilisateur" TYPE="STRING";
- COLUMN "LASTNAME" LABEL="Last name{F}Nom" TYPE="STRING";
- COLUMN "FIRSTNAME" LABEL="First name{F}Prénom" TYPE="STRING";
- COLUMN "REGIONNAME" LABEL="Region name{F}Nom région" TYPE="STRING";
- COLUMN "REGIONTYPE" LABEL="Region type{F}Type région" TYPE="STRING";
- COLUMN "READACCESS" LABEL="Read access{F}Accès lecture" TYPE="BOOLEAN";
- COLUMN "WRITEACCESS" LABEL="Write access{F}Accès écriture" TYPE="BOOLEAN";
- COLUMN "FRONTPCSYN" LABEL="FrontPC sychro{F}Synchro FrontPC" TYPE="BOOLEAN";

 Not available for eFront Cube programs.

### 4.1.16.2. Syntax

PROC EFRONTREGIONS </ options>;

RUN;

### 4.1.16.3. Options

Option	Description
USER =user_id	user_id: IQID of the user for whom to turn up the region optional;

CONNECTION= "named_connection"	named_connection: name of the database connection that allows the system to access another database.  Named connections must be defined previously (for details refer to the eFront Report Administrator Guide, <a href="#">Managing systematic access to external databases</a> ).  optional;
OUT=table	mandatory;  output table (ex.: WORK.REGIONS)

#### 4.1.16.4. Output

eFront Report table.

#### 4.1.16.5. Error messages

Message	Explanation

### 4.1.17. PROC EFRONTTABLES step

#### 4.1.17.1. Description

Independent program block. Always begins with the PROC EFRONTTABLES statement and ends with a RUN statement. Automatically generates **eFront Report** tables that describe the database data model.



Not available for eFront Cube programs.

#### 4.1.17.2. Syntax

PROC EFRONTTABLES (</ [table options](#)> </ options>);

RUN;

#### 4.1.17.3. Options

Option	Description
TABLES=table_name_1 table_name_n	Generates a table that describes the tables being specified
COLUMNS=col_name_1 col_name_n	Generates a table that describes the columns being specified

RELATIONS=rel_name_1 rel_name_n	Generates a table that describes the relations being specified
INDEXES=ind_name_1 ind_name_n	Generates a tables that describes the indexes being specified
INCLUDESYSTABLES	Include system tables in the output table

#### 4.1.17.4. Comment

Use PROC EFRONNTABLES to produce data model documentation.

#### 4.1.17.5. Output

No direct output.

#### 4.1.17.6. Error messages

Message	Explanation

### 4.1.18. PROC EFRONTUSERPROFILES step

#### 4.1.18.1. Description

Independent program block. Always begins with the PROC EFRONTUSERPROFILES statement and ends with a RUN statement. Retrieves the profiles of the users of the application.

Generates a table that includes 5 columns:

- COLUMN "USER\_ID" LABEL="User ID{F}ID utilisateur" TYPE="STRING";
- COLUMN "LOGIN" LABEL="User Login{F}Login utilisateur" TYPE="STRING";
- COLUMN "PROFILE\_ID" LABEL="Profile ID{F}Id profil" TYPE="STRING";
- COLUMN "PROFILENAME" LABEL="Name{F}Nom" TYPE="STRING";
- COLUMN "CURRENT" LABEL="Current profile{F}Profil courant" TYPE="BOOLEAN";

 Not available for eFront Cube programs.

#### 4.1.18.2. Syntax

PROC EFRONTUSERPROFILES </ options>;

RUN;

### 4.1.18.3. Options

Option	Description
USER =user_id	user_id: IQID of the user for whom to turn up the profile(s) optional;
OUT=table	mandatory; output table (ex.: WORK.USERPROFILES)

### 4.1.18.4. Output

eFront Report table.

### 4.1.18.5. Error messages

Message	Explanation

## 4.1.19. PROC ERROR step

### 4.1.19.1. Description

Independent program block. Always begins with the PROC ERROR statement and ends with a RUN statement. Stops the execution of a program by raising an exception with a custom error message.

### 4.1.19.2. Syntax

PROC ERROR MSG="Error\_message";

RUN;

### 4.1.19.3. Options

Option	Description
MSG="Error_message"	"Error_message"  An error message that displays when the program's execution is stopped.  Mandatory;

#### 4.1.19.4. Comment

- The PROC ERROR step and the ABORT option are similar in function, with the exception that ABORT must be part of the DATA step.

#### 4.1.19.5. Output

An error message that is triggered after the program's execution is stopped.

#### 4.1.19.6. Error messages

Message	Explanation

### 4.1.20. PROC EXPORTCHART step

#### 4.1.20.1. Description

Independent program block. Always begins with the PROC EXPORTCHART statement and ends with a RUN statement. Extracts all the charts from an EXCEL file and creates a JPEG file per chart.

 If you want to insert several charts into a WORD document, use the eFront Script step EXPORTCHART to extract all the charts from the EXCEL file into as many .JPEG files as there are charts, then use the WORD keyword IMG to insert all the charts into the WORD document.

 If you only want to extract and integrate one single chart from an EXCEL file into a WORD document, use the WORD keyword CHART.

 Not available for eFront Cube programs.

#### 4.1.20.2. Syntax

PROC EXPORTCHART FILE=Excel\_file\_name;

RUN;

### 4.1.20.3. Options

Option	Description
FILE=Excel_file_name	<p>Specifies the source EXCEL file. You need to indicate the absolute path to the file on your local server disk.</p> <p>Example:</p> <pre>FILE = "Excel_file.XLSX"</pre> <p>Make sure the IIS user on the web server is able to access the folder</p> <p>It is possible to build the file_name dynamically, and use macro variables to do so.</p> <p>You can also export a file to the DMS by specifying the name of the region and the name of the file.</p> <p> You have to specify the separator of the fields (TAB, COMMA, SEMICOLUMN, SPACE, PIPE).</p> <p>Example:</p> <pre>PROC EXPORT DATA=WORK.MATABLE FILE= "\My Private Region\Nom_de_fichier" SEPARATOR; RUN ;</pre>

### 4.1.20.4. Comment

- The step generates per chart, which is extracted from the EXCEL source file, a JPEG file with the name worksheet-chart.jpeg (like "Sheet1-Chart 1.jpeg").

### 4.1.20.5. Output

The step generates per chart, which is extracted from the EXCEL source file, a JPEG file with the name worksheet-chart.jpeg (like "Sheet1-Chart 1.jpeg").

## 4.1.21. PROC EXPORTEXCEL step

### 4.1.21.1. Description

Independent program block. Always begins with the PROC EXPORTEXCEL statement and ends with a RUN statement. Exports one or more **eFront Report** data tables to one or more EXCEL sheets of the same workbook (one table per sheet).

### 4.1.21.2. Syntax

PROC EXPORTEXCEL FILE=file\_name (</ file options>);

TABLE=table\_name\_1 (</ table options>);

...

TABLE=table\_name\_n (</ table options>);

RUN;

### 4.1.21.3. File options

Option	Description
FILE=file_name	<p>Specifies the path to the targeted EXCEL workbook on the client or the web server.</p> <p>You need to indicate the absolute path to the file + filename.</p> <p>Possible file extensions: XLS, XLSX</p> <p>Example:</p> <pre>FILE = "C:\efront\website\download\TABLE_ADRESSES.XLS"</pre> <p>Make sure that the folders referred to in the access path exists on your local server.</p> <p>It is possible to build the file_name dynamically using functions such as GETTEMPPATH(), store the result in macro variables</p> <p>Mandatory;</p>
DROP	<p>Specifies that the target file should be deleted before the export starts.</p> <p>If the DROP option is not activated, and if the target file already exists, the application adds the exported data as new EXCEL sheets.</p> <p>Optional;</p>
LABEL	Displays column labels in header (instead of column names)
NOHEADER	Specifies that no column titles are to be exported, no column titles in the first row
NODATA	Specifies that no data are to be exported, i.e.: only column titles are exported

### 4.1.21.4. Table options

Option	Description
--------	-------------

TABLE=table_to_export	Identifies the eFront Report data table to be exported.  Mandatory;
SHEETNAME=sheet_name	Specifies the name of the target sheet in the EXCEL workbook.  Mandatory;
LABEL	Displays column labels in header (instead of column names).  Optional;
NOHEADER	Specifies that no column titles are to be exported, no column titles in the first row.
NODATA	Specifies that no data are to be exported, i.e.: only column titles are exported.

#### 4.1.21.5. Comment

You can define the options NOHEADER, NODATA, LABEL at global level (first line of step) or at table level.

- PROC EXPORTEXCEL is an excellent alternative to the use of PROC PRINT if need to print tables with more than 1000 rows. If you use PROC PRINT to print tables with more than 1000 rows, the system might crash the **eFront** application or the internet browser.

#### 4.1.21.6. Output

Data table written to a sheet of an EXCEL workbook.

#### 4.1.21.7. Error messages

Message	Explanation

### 4.1.22. PROC EXPORT step

#### 4.1.22.1. Description

Independent program block. Always begins with the PROC EXPORT statement and ends with a RUN statement. Exports an **eFront Report** data table or view formatted as **.CSV** file to your local server disk or to the DMS by giving a file name. You can export the table structure only or the table structure including the data.

## 4.1.22.2. Syntax

PROC EXPORT DATA=data\_to\_export (</ table\_options> </ options>);

RUN;

## 4.1.22.3. Options

Option	Description
COMMA	Specifies that the separator to use in the .CSV file is a comma (CSV: Comma Separated Values).
DATA=data_to_export	Identifies the data to export which can be an eFront Report data table or an eFront Report view.
DROP	<p>Specifies that the target file should be deleted before the export starts.</p> <p>If the DROP option is not activated, and if the target file already exists, the application adds the exported data as new lines at the end of the existing file.</p> <p>There is no automatic verification comparing the column structure of the already existing file with the column structure defined for the data table to be exported and appended to the existing file.</p>
FILE=file_name	<p>Specifies the target file. You need to indicate the absolute path to the file on your local server disk.</p> <p>Example:</p> <pre>FILE = "C:\efront\website\download\TABLE_ADRESSES.CSV"</pre> <p>Make sure that the folders referred to in the access path exists on your local server.</p> <p>It is possible to build the file_name dynamically, and use macro variables to do so.</p> <p>You can also export a file to the DMS by specifying the name of the region and the name of the file.</p> <p> You have to specify the separator of the fields (TAB, COMMA, SEMICOLUMN, SPACE, PIPE).</p> <p>Example:</p> <pre>PROC EXPORT DATA=WORK.MATABLE FILE= "\My Private Region\Nom_de_fichier" SEPARATOR; RUN ;</pre>
LABEL	Exports column labels as column titles.
NOHEADER	Specifies that no column titles are to be exported.
NODATA	Specifies that no data are to be exported, i.e.: only column titles are exported.

NOERROR	Specifies that export is to be continued even if an error occurs. For example, if the DROP option doesn't succeed in deleting the target file.
OUT	Specify the output data table of the export.
SEMICOLON	Specifies that the separator to use in the .CSV file is a semicolon.  If you don't specify any separator, the semicolon is the separator used by default.
SPACE	Specifies that the separator to use in the .CSV file is a space.
TAB	Specifies that the separator to use in the .CSV file is a tab.
FORCEQUOTE	Wraps double quotes around any character string that contains a separator or double quote.
ENCODING	Defines the encoding of the exported file. Can be defined as follows: <ul style="list-style-type: none"> <li>• with the code page name (e.g. ENCODING="us-ascii")</li> <li>• with the code page identifier (e.g. ENCODING="20127")</li> </ul>

#### 4.1.22.4. Comment

- For the moment there is no escape character being specified that would allow eFront Script to interpret correctly separators being included in any character string of the original data table.
- Select the separator TAB, if you want to export in **UNICODE** to make sure that data table columns are correctly transferred into .CSV file columns.
- There is no verification as to the possible combinations of options.

#### 4.1.22.5. Output

.CSV file

#### 4.1.22.6. Error messages

Message	Explanation
Export file cannot be generated : System.IO.DirectoryNotFoundException: Could not find a part of the path 'C:\efront\website\download\TABLE_ADRESSES.CSV'.	The export did not succeed, because the folder structure that should receive the .CSV file doesn't exist.

## 4.1.23. PROC FALIBRARY step

### 4.1.23.1. Description

Independent program block. Always begins with the PROC FALIBRARY statement and ends with a RUN statement. Imports an **eFront Analytics** standard library into **eFront Report**.

### 4.1.23.2. Syntax to be used to call a standard library

```
PROC FALIBRARY OUT=Output_table TABLE="standard_FA_library"
(STDLib_Param_1=Parameter_value_1 STDLib_Param_n=Parameter_value_n);
```

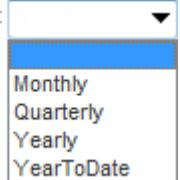
RUN;

### 4.1.23.3. OPTIONS to be used together with a call to a standard library

Option	Description
OUT=Output_table	Refers to the target eFront Report output table
TABLE="standard_FA_library"	Refers to the standard FA library to be used as data source eFront Analytics data table / data cube
STDLib_Param_n=Parameter_value	<p>Refers to the parameters to be passed along with the standard eFront Analytics library</p> <p>Example:</p> <p>(IQID=%IQID TODATE=TODAY())</p> <p>Note: You need to use the internal / program name of the parameter!</p>

### 4.1.23.4. Parameters to be passed along with standard libraries

Parameter	Technical name	Type of field	
Portfolio	IQID	Search	
Company		Example:	
Direct Investor		<input type="text" value="Fund:"/> 	
Fund Investor			
Property			
Deal			
Fund			

Next step start date	FROMDATE	Calendar	
Next step end date	TODATE	Example:	
Reference Date		Reference date: <input type="text"/> 	
Investee	INVESTEE	List	
Period	PERIOD	Example:	
Dashboard currency	REFCURRENCY	Period: 	
Reporting Currency	CURRENCY		
Reference surface unit	REFUNIT		

Syntax to be used to call a specific library

```
PROC FALIBRARY OUT=Output_table TABLE="path_to_specific_library"
(SpecificLib_Param_1=Parameter_value_1 SpecificLib_Param_n=Parameter_value_n);
```

RUN;

#### 4.1.23.5. OPTIONS to be used together with a call to a specific library

Option	Description
OUT=Output_table	Refers to the target eFront Report output table
TABLE="path_to_specific_library"	Refers to the path to the specific eFront Analytics library. The path is composed of the folder name, followed by the library name.  Example:  TABLE="Dashboard Folder\S_PEFundAssets.MainInfos(PORTFOLIO)"
SpecificLib_Param_n=Parameter_value	Refers to the parameters to be passed along with the specific eFront Analytics library  Example:  (IQID=%IQID TODATE=TODAY() REFCURRENCY=%DASHBOARD_CURRENCY)  Note: You need to use the internal / program name of the parameter!

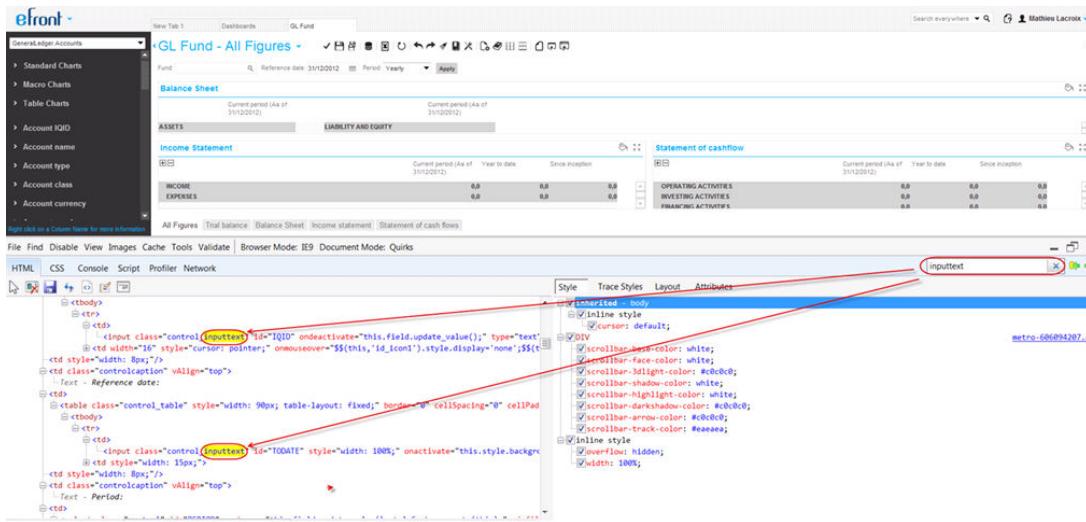
## 4.1.23.6. Comment

- The key to using the FALIBRARY step correctly is to find the parameters to be passed along with the library. When you refer to the parameters to be passed along with the library, you need to use the technical (internal names) of the parameters. Get a **Tip: How to get the internal names of library parameters.**

### Prerequisite:

- You use IE as browser.

- Open a dashboard, which uses the library you want to call from the FALIBRARY step.
- Access the browser's developer tool:
  - Click F12.
 The developer's tool opens.
- Search for technical parameter names:
  - If the parameter expects text to be entered, do a research on **inputtext**.



The property **Id** is set to the internal name of the parameter.

- If the parameter expects a list item to be selected, do a research on the value of the selected item.

The screenshot shows the eFront interface for 'GL Fund - All Figures'. In the top right, there is a dropdown menu set to 'Yearly'. A red arrow points from this dropdown to the 'yearly' option in the list of options. Below the interface, the browser's developer tools are open, specifically the 'Elements' tab. The code pane shows a section of HTML with a dropdown menu. One of the options, 'Yearly', has a red circle around its 'value' attribute. A second red arrow points from this circled 'value' attribute to the same 'value' attribute in the developer tools' element inspector.

The property **Id** is set to the internal name of the parameter.  
Here is another example:

The screenshot shows the eFront interface for 'My Dashboard - Page'. In the top right, there is a dropdown menu labeled 'Reference surface unit'. A red arrow points from this dropdown to the 'REFUNIT' option in the list of options. Below the interface, the browser's developer tools are open, specifically the 'Elements' tab. The code pane shows a section of HTML with a dropdown menu. One of the options, 'Reference surface unit', has a red circle around its 'value' attribute. A second red arrow points from this circled 'value' attribute to the same 'value' attribute in the developer tools' element inspector.

- For more details about the parameters to be used with each library, please refer to the **eFront Analytics - Catalogue of Standard Libraries and Sample Dashboards**.

#### 4.1.23.7. Output

eFront Report data table.

#### 4.1.23.8. Error messages

Message	Explanation
---------	-------------

### 4.1.24. PROC FAQUERY step

#### 4.1.24.1. Description

Independent program block. Always begins with the PROC FAQUERY statement and ends with a RUN statement. Exports data from a **QueryBuilder** query into an **eFront Report** table.

#### 4.1.24.2. Syntax

PROC FAQUERY </ options>;

</ subordinate statements>;

RUN;

#### 4.1.24.3. Options

Option	Description
--------	-------------

QUERY="query_access"(param_name="value")	"query_access": Refers to the access path to the query component. Mandatory; Ex.: QUERY="DEAL\testqb"; Note: 'DEAL' refers to the Dashboard Folder that hosts the query component. (param_name="value"): You need to pass along the list of all the parameters required by the query. For each parameter you need to indicate the name of the parameter and its value. Optional; Ex.: QUERY="DEAL\testqb"(IQID="0F46B969A46A45F88C921CF55F27C357");
--	---

#### 4.1.24.4.

#### 4.1.24.5. Subordinate statements

Option	Description
--------	-------------

<pre>TABLE "query_name"(COLUMNS=COL1 COLx) OUT=Output_table;</pre>	<p><b>"query_name":</b></p> <p>Refers to the query name as defined in the query component.</p> <p>Note: If the query component has only one single query, its name is set to "default".</p> <p>Remember: several queries can be linked to a query component!</p> <p>Mandatory;</p> <p>You can use <a href="#">macro_variables</a> instead of the explicit query name!</p> <p>(COLUMNS=COL1 COLx):</p> <p>Refers to the columns of the query to be transmitted. By default, the step transmits all of the columns.</p> <p>COL1 refers to the first column,</p> <p>COL2 refers to the second column,</p> <p>etc.</p> <p>Note: Column names must meet the requirements listed in the <a href="#">▶ Naming columns</a> section.</p> <p>Column names must meet the following requirements:</p> <ul style="list-style-type: none"> <li>• can contain only numbers, plain letters or underscores (no accents).</li> <li>• cannot start with a number, except if the name is enclosed within square brackets.</li> <li>• cannot contain any blank spaces within the name, except if the name is enclosed within square brackets.</li> <li>• max length: 255 char.</li> </ul> <p>Optional;</p> <p>OUT=Output_table:</p> <p>Refers to the target eFront Report output table</p> <p>Mandatory;</p> <p>Examples:</p> <p>TABLE "default"(COLUMNS=col2 col4) OUT=work.test</p>
--	--

#### 4.1.24.6. Comment

- You can use more than one TABLE statement in a PROC FAQUERY step.
- To get the correct name of the parameters to be used in the PROC FAQUERY, you need to check the parameter's name as defined in the QueryBuilder query **Properties**.

#### 4.1.24.7. Output

eFront Report data table.

#### 4.1.24.8. Error messages

Message	Explanation
---------	-------------

### 4.1.25. PROC FORMAT step

#### 4.1.25.1. Description

Independent program block. Always begins with the PROC FORMAT statement and ends with a RUN statement. Creates customized formats for data display. You can format character strings or numeric values. Customized formats can be referred to using the **FORMAT** statement and the **FORMAT** option. Formats manage the transition from value storage to value display. Formats replace values as being stored by values as being displayed.

#### 4.1.25.2. Syntax

To define a format that applies to **character strings**, use the following syntax :

```
PROC FORMAT </ options>;
  VALUE $format_name
    value_or_range_1 = "value_displayed_1"
    value_or_range-n = "value_displayed_n"
  ;
RUN;
```

To define a format that applies to **numeric values**, use the following syntax :

```
PROC FORMAT <options>;
```

```
PICTUREpicture_name
```

```
value_or_range_1 = "picture"
```

```
value_or_range_n = "picture"
```

```
;
```

```
RUN;
```

#### 4.1.25.3. Options

Option	Description
LIB=format_library	Relative or absolute path to format library. If you don't specify a format_library , eFront Script stores formats temporarily in the WORK library. In this case, formats only exist during the eFront Script work session.

#### 4.1.25.4. Comment

- Formats apply to values. Formats allow to replace one value by another value. Therefore, when defining a format, you must be able to identify the values to replace.
- The **VALUE** or **PICTURE** statement is required. You can use several VALUE or PICTURE statements in a PROC FORMAT step.
- When referencing a format from within a programming step, set a period at the end of the format, ex: format.

#### 4.1.25.5. Output

No direct output.

#### 4.1.25.6. Error messages

Message	Explanation
---------	-------------

## 4.1.26. PROC FUNDCASHFLOWS step

### 4.1.26.1. Description

Independent program block. Always begins with the PROC FUNDCASHFLOWS statement and ends with a RUN statement. Generates an eFront Report data table that returns all fund cash flows (fund operation IDs, or fund operation IDs by share if shares are managed) for a collection of fund share IDs and/or fund operation IDs. The data is directly extracted from the database via the **Calculation Engine (FCE)**.

PROC FUNDCASHFLOWS returns a list of fund cash flows - information about positions is not included! The advantage of using the table produced by the eFront Script step lies in increased performance in extracting information.

 If you need a fund cash flow table, which also contains the positions, we recommend you use the table [FCE.FUNDCASHFLOWS](#). If you do so, you don't need to use the PROC FUNDCASHFLOWS.

 For each investor, investor-by-share, the step generates a "ghost" operation. This operation, which is an adjustment, enables the calculation of investor, investor-by-share positions by adding up column content, even for values, which depend on FX variations such as the valuation and residual commitment.

 Not available for eFront Cube programs.

### 4.1.26.2. Syntax

PROC FUNDCASHFLOWS </ options>;

RUN;

### 4.1.26.3. Options

Option	Mandatory?	Description
--------	------------	-------------

SUBSCRFUNDOP=subscrfundop	One of these options is mandatory.	subscrfundop: --> a collection of ID's of subscriber fund operations (table: VCSUBSCRFUNDOP) or --> a SQL query, such as 'SELECT IQID from...'
SUBSCRFUNDOPSHARE=subscrfundopshare		subscrfundopshare: --> a collection of ID's of subscriber fund operation shares (table: VCSUBSCRFUNDOPSHARE) or --> a SQL query, such as 'SELECT IQID from...'
OUT=lib.table		mandatory; output table

#### 4.1.26.4. Comment

Depending on the selected option, PROC FUNDCASHFLOWS returns:

- Fund operations per investor.
- Fund operations per investor and per share.

The ID of the generated ghost operation corresponds to the Investor / Share ID.

#### 4.1.26.5. Applicable macros

You can modify the default behavior of PROC FUNDCASHFLOWS using the following macros:

- If you define the %REPORT\_DATE macro, the date to be used is the value of the variable %DATE, otherwise the system uses the current date.
- If you define the %USE\_DRAFT, fund draft operations are used by the calculation engine, otherwise they are ignored.
- If you define the %USE\_INVESTOR\_CURR, calculations are done in investor currency, otherwise calculations are done in fund currency.

- If you define the %START\_DATE, the system will take the value defined for the start date, and perform calculations for all the fund operations up from this date.

#### 4.1.26.6. Output

Generates an eFront Report data table.

#### 4.1.26.7. Error messages

Message	Explanation
---------	-------------

### 4.1.27. PROC GCHART step

#### 4.1.27.1. Description

Independent program block. Always begins with the PROC GCHART statement and ends with a RUN statement. Creates different types of charts from an **eFront Report** data table, such as pie charts, horizontal bars' charts, vertical bars' charts, and donuts.

 Not available for eFront Cube programs.

#### 4.1.27.2. Syntax

```
PROC GCHART DATA = table_to_process (</ table_options> </ options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

#### 4.1.27.3. Options

Option	Description
SORTDESC	Sorts values in descending mode.
STYLE=stylesheet_name	Applies an existing stylesheet to PROC GCHART items. The stylesheet definition must precede the PROC GCHART step.
STYLE(item_to_be_styled) = (</ chart_style_attributes>)	Applies a particular style to PROC GCHART item. Refer to the <b>STYLESHEET step</b> topic to identify GCHART items that can be styled.

#### 4.1.27.4. Subordinate statements

Statement	Description
DONUT	Specifies the column(s) to include in the donut chart
DONUT3D	Specifies the column(s) to include in the 3-dim donut chart
HBAR	Specifies the column(s) to include in the horizontal bars' chart
HBAR3D	Specifies the column(s) to include in the horizontal 3-dim bars' chart
PIE	Specifies the column(s) to include in the pie chart
PIE3D	Specifies the column(s) to include in the pie chart
TITLE	Specifies the table title. 10 hierarchical levels are available
VBAR	Specifies the column(s) to include in the vertical bars' chart
VBAR3D	Specifies the column(s) to include in the vertical 3-dim bars' chart

- A subordinate statement that defines the chart type is required.
- If labels overlap on the x-axis, modify the chart width, or choose an different x-axis display angle. Example:  

```
PROC GCHART DATA = table_to_processSTYLE=(WIDTH=450);
PROC GCHART DATA = table_to_processSTYLE=(XAXISANGLE=20);
```

#### 4.1.27.5. Output

Chart.

#### 4.1.27.6. Error messages

Message	Explanation
ERROR: Chart type not defined	The PIE statement or PIE3D statement has been left out.

### 4.1.28. PROC GPLOT step

#### 4.1.28.1. Description

Independent program block. Always begins with the PROC GPLOT statement and ends with a RUN statement. Creates plot and line charts from an **eFront Report** data table.

 Only available for eFront Cube programs.

## 4.1.28.2. Syntax

```
PROC GPLOT DATA = table_to_process (</ table_options> </ options>);

</ subordinate statements>

RUN;
```

## 4.1.28.3. Options

Option	Description
<code>STYLE=stylesheet_name</code>	Applies an existing stylesheet to PROC GPLOT items. The stylesheet definition must precede the PROC GPLOT step.
<code>STYLE = (&lt;/graph_style_attributes&gt;)</code>	Applies a particular style to PROC GPLOT items.

## 4.1.28.4. Subordinate statements

Statement	Description
<code>PLOT</code>	Specifies the column(s) to include in the plots & lines chart
<code>TITLE</code>	Specifies the table title. 10 hierarchical levels are available.

- The PLOT statement is required.
- If labels overlap on the x-axis, modify the chart width, or choose a different x-axis display angle. Example:

```
PROC GPLOT DATA = table_to_process STYLE=(WIDTH=450);
PROC GPLOT DATA = table_to_process STYLE=(XAXISANGLE=20);
```

## 4.1.28.5. Output

Plots & lines chart.

## 4.1.28.6. Error messages

Message	Explanation
<code>ERROR: Chart type not defined</code>	The PIE statement or PIE3D statement has been left out.

## 4.1.29. PROC MEMORY step

### 4.1.29.1. Description

Independent program block. Always begins with the PROC MEMORY statement and ends with a RUN statement. Displays data table size and unloads data tables from memory.

### 4.1.29.2. Syntax

PROC MEMORY (</options>);

RUN;

### 4.1.29.3. Options

Option	Description
UNLOAD	Unloads the folder/table being defined. If no folder/table is specified, all the tables are unloaded.  If PROC MEMORY is used without UNLOAD, the tables being loaded into memory show in the LOG window, together with the space taken up.
NOLIST	Avoids listing tables in memory
LIBRARY=folder_name	Defines the folder whose tables are to be unloaded/or whose memory properties are to be listed in the LOG window.  E.g.: LIBRARY=WORK
DATA=table_name	Defines the table to be unloaded or whose memory properties are to be listed in the LOG window.

### 4.1.29.4. Comment

- By default, **eFront Report** unloads data tables after **program execution** (PROC MEMORY UNLOAD NOLIST). If you need tables to remain loaded, especially during development, open **Administration > My options**, and select **No** under **Unload tables**. We recommend you do not change the **Unload tables** option in **Administration > System options**.

### 4.1.29.5. Output

LOG window.

#### 4.1.29.6. Error messages

Message	Explanation

#### 4.1.30. PROC MEANS step

##### 4.1.30.1. Description

Independent program block. Always begins with the PROC MEANS statement and ends with a RUN statement. Calculates statistics for an **eFront Report** data table.

##### 4.1.30.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) </ options>;
```

```
</ subordinate statements>;
```

```
RUN;
```

##### 4.1.30.3. Options

Option	Description
OUT = output_table	Identifies the output_table. If you don't specify an output_table, eFront Script overwrites the table_to_process
MISSING	When applying the PROC MEANS step, eFront Script omits missing values by default. If you specify the MISSING option, the procedure treats missing values of a categorical variable as a valid category.

##### 4.1.30.4. Subordinate statements and statistical functions

Option	Description
BY	Identifies the data groups for which to calculate statistics.
CLASS	Selects the values for which statistics are to be calculated.
FIRST	Displays the first value of a list of values.
GEOMETRICMEAN	Returns the geometric mean.
HARMONICMEAN	Returns the harmonic mean.
IRR	Calculates the Internal Rate of Return.
JOIN	Groups a list of values in a single column cell.

KURTOSIS	Returns the kurtosis.
LAST	Displays the last value of a list of values.
MAX	Returns the maximum value of the values of a column.
MEAN	Returns the average of the values of a column.
MEDIAN	Returns the median.
MIN	Returns the minimum value of the values of a column.
MULTIPLE	Returns the multiple of the return on investment.
N	Total number of variable instances.
NOBS	Calculates the number of rows in each aggregated category .
NONE	Returns 'None'.
PCTN	Returns the PCTN (number of aggregated rows/total number of rows).
PCTSUM	Returns the PCTSUM (sum of the aggregated rows/sum of the total number of rows).
RANGE	Returns the range.
SKEWNESS	Returns the skewness.
STDDEV	Returns the standard deviation.
SUM	Calculates the sum of a column.
SUMNEG	Returns the negative sum.
SUMPOS	Returns the positive sum.
VAR	Selects the variables to be analyzed.
VARIANCE	Returns the variance.

#### 4.1.30.5.

#### 4.1.30.6. Comment

- The statistical analysis concerns only the sub-group of data that remain after possible data filters have been applied to the table\_to\_process. The table\_to\_process itself is not modified.

#### 4.1.30.7. Output on screen

No direct output.

#### 4.1.30.8. Error messages

Message	Explanation

## 4.1.31. PROC OFFICE step

### 4.1.31.1. Description

Independent program block. Always begins with the PROC OFFICE statement and ends with a RUN statement. Creates a **.DOC**, **.XLS**, **.PDF**, or **XML** file by using data from one or more **eFront Report** data tables.

Note that you can also use the **eFront Report** editor to produce .DOC and .XLS documents. By contrast, XML files are not supported in the eFront Report editor and can only be generated with eFront Script programs.

### 4.1.31.2. Syntax

PROC OFFICE FILE=file\_name;

SETtable\_name\_1 (</ table options>) table\_name\_n (</ table options>);

TEMPLATE&file\_name;

RUN;

### 4.1.31.3. Options

Option	Description
FILE=file_name	The name and path of the file to be created. The file path can be relative or absolute, for example: FILE= "OutputFile.xls" FILE="Export.xml" FILE="\Folder-1\...Folder-n\OutputFile.xls"

### 4.1.31.4. Subordinate statements

Statement	Description
-----------	-------------

<b>TEMPLATE</b>	<p>Defines the template (XLS, DOC, XML) to be used to produce the final report. The template needs to be attached to the program by uploading it in the <b>Attachments</b> section on the left-hand pane of the program page.</p> <p>Note that the attached template must be prefaced with an ampersand (&amp;) in the script, for example:</p> <pre>TEMPLATE "&amp;Template.xls";</pre>
<b>SET</b>	<p>Defines the tables whose data are to be used to feed the report. Reads all the lines of 1 or more existing tables into the output table (concatenation of tables)</p>

#### 4.1.31.5.

#### 4.1.31.6. Comment

- To ensure that the template is recognized by the program, prefix the template's name with "&" ("&TestTemplate.xml").
- **SET** and **TEMPLATE** statements are required.
- You can also use .XLS and .DOC reports in the eFront Report document editor in order to generate .XLS and .DOC reports.
- At present, the eFront Report document editor doesn't accept XML templates. You can only generate XML reports with eFront Script programs.
- PDF files can only be generated with eFront Script programs. eFront Cube programs currently do not support PDF generation.

#### 4.1.31.7. Output

Generates an **.XLS**, **.DOC**, **.PDF** or **XML** file. The XML template used to produce the XML file is a valid XML file whose nodes are interpreted by the server in order to produce the final XML report. For more information, refer to the **TEMPLATE** statement.

#### 4.1.31.8. Error messages

Message	Explanation

### 4.1.32. PROC PRINT step

#### 4.1.32.1. Description

Independent program block. Always begins with the PROC PRINT statement and ends with a RUN statement. The PRINT step prints an **eFront Report** data table, including all

the data or only part of it. You can create a variety of reports ranging from simple listings to customized reports that group the data and calculate totals and subtotals for numeric variables. Note that the PROC PRINT step has no influence on the database tables.

#### 4.1.32.2. Syntax 1

```
PROC PRINT DATA = table_to_print (</ table_options> </ options>);

</ subordinate statements>

RUN;
```

#### 4.1.32.3. Syntax 2

```
PROC PRINT;

PUT <value_to_print>;
```

```
RUN;
```

#### 4.1.32.4. Options

Option	Description
<b>FREEZEHEADER</b>	Freezes the column headers, so that the user can scroll through the report without losing the column headers. We recommend to use FREEZEHEADER together with FREEZELEFT.
<b>FREEZELEFT=&lt;n&gt;</b>	<n>: integer.  Freezes left-hand columns from horizontal scrolling. We recommend to use FREEZEHEADER together with FREEZELEFT.
<b>LABEL</b>	Specifies that column labels display as column titles
<b>NOGRANDTOTAL</b>	
<b>NOOBS</b>	Removes the first column from the output table (eFront Script adds automatically a first column to each output table that lists line numbers)
<b>N=&lt;n&gt;</b>	<n>: string  Name of the Total column
<b>STYLE=stylesheet_name</b>	Applies an existing stylesheet to PROC PRINT items. The stylesheet definition must precede the PROC PRINT step.
<b>STYLE(item_to_be_styled) = (table_style_attributes)</b>	Applies a particular style to a PROC PRINT item.

<b>URL=</b>	Specifies the link that allows you to access the variable value from within a table.
-------------	--

#### 4.1.32.5. Subordinate Statements

Statement	Description
<b>BY</b>	Produces a separate section of the report for each BY group.  You can choose how many groups will be opened (Expand =... ) or closed (Collapse = ...) when using the BY statement.
<b>EXPORT</b>	Exports the table to the location being specified
<b>FORMAT</b>	Applies a format to one or more columns
<b>PUT</b>	Writes data values to a special buffer from which they can be written to the data component.
<b>SUM</b>	Calculates the total of numeric values
<b>TITLE</b>	Defines the table title to display. 10 hierarchical levels are available.
<b>VAR</b>	Identifies the variables to be printed and determines the order of appearance. Print order = list order in statement

#### 4.1.32.6. Embeddable steps

Step	Description
<b>EVENT</b>	Allows you to embed a control structure within the PROC PRINT step, which is not possible otherwise. Embedding a control structure might be of interest if you want to apply conditional cell coloring for example. (View an example: <a href="#">PROC PRINT - EVENT PREDATAROW</a> )
<b>TABLE</b>	Allows you to print data tables in the different cells of a table.

#### 4.1.32.7. Comment

- By default, **eFront Script** adds a first column (column name = OBS) to each table\_to\_print. OBS lists table line numbers.
- By default, **eFront Script** displays column **names** as column titles. To display column **labels**, use the LABEL option.
- You can use PROC PRINT without specifying the table\_to\_print: **eFront Script** prints the last table that has been created during the **eFront Script** work session.
- You can use PROC PRINT as a means to write text to the output. In this case, you don't need to indicate a data table.
- If you want to print data tables with more than 1000 rows, consider using the PROC EXPORTEXCEL to avoid crashing the Internet Browser or your eFront application.

- Note that any filters used in the PROC PRINT step will not function in eFront Cube programs. Filters for the PROC PRINT step are currently supported only in eFront Script programs.
- PROC PRINT in eFront Cube programs can only print 100 rows.

#### 4.1.32.8. Output

Listing (HTML output).

#### 4.1.32.9. Error messages

Message	Explanation
ERROR: Invalid proc option	One of the procedure options is invalid.

### 4.1.33. PROC PRINTCOL step

#### 4.1.33.1. Description

Independent program block. Always starts with the PROC PRINTCOL statement and ends with a RUN statement. Writes a column of a data table into a multi-column print table.

 Not available for eFront Cube programs.

#### 4.1.33.2. Syntax

```
PROC PRINTCOL DATA = table_to_print (</ table_options> </ options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

#### 4.1.33.3. Options

Option	Description
--------	-------------

Flow=	Defines the way of writing data table values into the print table columns. Values can be written per line (horizontally) or per column (vertically).  Possible values: HORIZONTALLY   VERTICALLY  Default value: VERTICALLY
COLUMNS=integer	Defines the number of columns in the printed table
STYLE(DATA)=(WIDTH=integer)	Defines the width of the printed table columns
STYLE=stylesheet_name	Applies an existing stylesheet to PROC PRINTCOL items. The stylesheet definition must precede the PROC PRINTCOL step.
STYLE(item_to_be_styled) = (table_style_attributes)	Applies a particular style to a PROC PRINTCOL item.
URL=	Specifies the link that allows to access the variable value from within a table.

#### 4.1.33.4. Subordinate Statements

Statement	Description
TITLE	Defines the table title to display. 10 hierarchical levels are available.
VAR	Identifies the variable to be printed. You can only print one variable.

#### 4.1.33.5. Comment

- At least one data table column must be display per print table line.

#### 4.1.33.6. Output

Listing (HTML output).

#### 4.1.33.7. Error messages

Message	Explanation

### 4.1.34. PROC PRINTFORM step

#### 4.1.34.1. Description

Independent program block. Always begins with the PROC PRINTFORM statement and ends with a RUN statement. Allows you to print a form.

 Not available for eFront Cube programs.

#### 4.1.34.2. Syntax

```
PROC PRINTFORM DATA = table_to_print (</ table_options> </ options>);

</ subordinate statements>

RUN;
```

#### 4.1.34.3. Options

Option	Description
<a href="#">FREEZEHEADER</a>	Freezes the column headers, so that the user can scroll through the report without loosing the column headers. We recommend to use FREEZEHEADER together with FREEZELEFT.
<a href="#">FREEZELEFT=&lt;n&gt;</a>	<n>: integer.  Freezes left-hand columns from horizontal scrolling. We recommend to use FREEZEHEADER together with FREEZELEFT.
<a href="#">LABEL</a>	Specifies that column labels display as column titles
<a href="#">NOGRANDTOTAL</a>	
<a href="#">NOOBS</a>	Removes the first column from the output table (eFront Script adds automatically a first column to each output table that lists line numbers)
<a href="#">N=&lt;n&gt;</a>	<n>: integer  Returns the total of table lines
<a href="#">STYLE=stylesheet_name</a>	Applies an existing stylesheet to PROC PRINT items. The stylesheet definition must precede the PROC PRINT step.
<a href="#">STYLE(item_to_be_styled) = (table_style_attributes)</a>	Applies a particular style to a PROC PRINT item.
<a href="#">URL=</a>	Specifies the link that allows to access the variable value from within a table.

#### 4.1.34.4. Subordinate Statements

Statement	Description
<a href="#">BY</a>	Produces a separate section of the report for each BY group
<a href="#">EXPORT</a>	Exports the table to the location being specified

<b>EVENT</b>	Allows you to embed a control structure within the PROC PRINT step, which is not possible otherwise. Embedding a control structure might be of interest if you want to apply conditional cell coloring for example. (View an example: <a href="#">PROC PRINT - EVENT PREDATAROW</a> )
<b>FORMAT</b>	Applies a format to one or more columns
<b>PUT</b>	Writes data values to a special buffer from which they can be written to the data component.
<b>SUM</b>	Calculates the total of numeric values
<b>TITLE</b>	Defines the table title to display. 10 hierarchical levels are available.
<b>VAR</b>	Identifies the variables to be printed and determines the order of appearance. Print order = list order in statement

#### 4.1.34.5. Comment

- You can use the function **TABLENOBS()** to test the number of lines of the table to print.
- By default, **eFront Script** displays column **names** as column titles. To display column **labels**, use the **LABEL** option.

#### 4.1.34.6. Output

Listing (HTML output).

#### 4.1.34.7. Error messages

Message	Explanation
<a href="#">ERROR: Invalid proc option</a>	One of the procedure options is invalid.

### 4.1.35. PROC SENDTOIC step

#### 4.1.35.1. Description

Independent program block. Always begins with the PROC SENDTOIC statement and ends with a RUN statement. Sends documents (PDF, DOC, DOCX) from **eFront Invest to Investment Café Classic**.

 Not available for eFront Cube programs.

#### 4.1.35.2. Syntax

%LET BATCH=null;

PROC SENDTOIC </ options>;

RUN;

#### 4.1.35.3. Options

Option	Description
FILE=FI_file_name	<p>Relative or absolute path to file. Ex.:</p> <p>FILE= "FileToBeSent.pdf"</p> <p>FILE="FileToBeSent.docx"</p> <p>FILE="\Folder-1\...Folder-n\FileToBeSent.pdf"</p> <p>Transmittable formats:</p> <p>.DOC</p> <p>.DOCX</p> <p>.PDF</p> <p>Mandatory option</p>
TITLE=IC_file_name	<p>IC_file_name:</p> <p>Refers to the filename under which the file will be integrated into Investment Café Classic. This must contain the proper file extensions (for example: Call_2014_Q1.docx). Note: Using this parameter, your file can be stored in FI as \\somepath\A.docx, and yet be downloadable from Investment Café Classic as "B.docx"</p> <p>Mandatory option</p>
FUND=fund_IQID	<p>fund_IQID:</p> <p>Refers to the IQID of the fund to which the document to be sent is linked to</p> <p>Mandatory option</p>
INVESTOR=investor_IQID	<p>investor_IQID:</p> <p>Refers to the IQID of the investor, who is part of the fund's distribution list, and who is defined as target of the document to be sent</p> <p>Mandatory option</p>

TYPE=document_type	<p>document_type:</p> <p>Refers of the eFront Invest document type linked to the document to be sent.</p> <p>Document types are defined in the reference table: AJXPENOTICETYPE</p> <p>Note: The eFront Invest document type must be mapped to an Investment Café Classic document type. This is done from within eFront Invest: Misc &gt; Investment Café Classic &gt; Category Mappings</p> <p>Mandatory option</p>
NOTICENAME=document_description	<p>document_description:</p> <p>Refers to a short description of the document to be sent</p> <p>Mandatory option</p>
DATE=date	<p>date:</p> <p>Refers to the reference date for the document to be sent</p> <p>Mandatory option</p>
BATCH=batch	<p>batch:</p> <p>Implements the concept of batch on Investment Café Classic. You must set the batch initially to null. This parameter works by reference (not by value): the first call to SENDTOIC will set it with a batch number (since it is initially null), and then you can pass it further to other SENDTOIC calls.</p> <p>Hence, you can keep "adding" documents to the same batch again and again. Note that the actual upload is done file-by-file every time that SENDTOIC is called, they are part of the same Investment Café Classic batch.</p> <p>Optional parameter</p>

#### 4.1.35.4. Comment

- When sending fund notices from eFront Invest to a linked Investment Café Classic, users should use the **Send to IC** button which is available from within the fund notice generator. The PROC SENDTOIC step should only be used in special cases.

#### 4.1.35.5. Output

Document sent from eFront Invest to Investment Café Classic.

#### 4.1.35.6. Error messages

Message	Explanation
---------	-------------

#### 4.1.36. PROC SETFILTER step

##### 4.1.36.1. Description

Independent program block. Always begins with the PROC SETFILTER statement and ends with a RUN statement. Filters automatically an eFront Cube cube according to the defined filtering expressions, at the moment when it is accessed by a dashboard.

 Applies only to eFront Cube programs.

##### 4.1.36.2. Syntax

PROC SETFILTER </ options>;

RUN;

##### 4.1.36.3. Options

Option	Description
DATA=cube	<p>cube: name of cube generated by the eFront Cube program</p> <p>Example:</p> <pre>DATA = SESSION.T_Dashboard_Main_Table</pre> <p>mandatory;</p>
FILTER=(filteringExpression_1 filteringExpression_n)	<p>(filteringExpression_x): expression(s) used to filter the cube</p> <p>Examples:</p> <pre>FILTER= (DATE &lt;= %TODATE)</pre> <pre>FILTER= (DATE &lt;= %TODATE PORTFOLIO=%allPortfolios)</pre> <p>mandatory;</p>

#### 4.1.36.4. Comment

- You can use as many filtering expressions as necessary.
- You use typically the PROC SETFILTER step in eFront Cube programs which are directly linked to dashboards and used to filter global cubes to make sure that only needed data are uploaded from the eFront Cube server.

#### 4.1.36.5. Output

Filtered cube related to a dashboard.

#### 4.1.36.6. Error messages

Message	Explanation

### 4.1.37. PROC SORT step

#### 4.1.37.1. Description

Independent program block. Always begins with the PROC SORT statement and ends with a RUN statement. Sorts tables by the values of one or more character or numeric variables. The PROC SORT step either replaces the original table or produces a new one.

#### 4.1.37.2. Syntax

PROC SORT DATA = table\_to\_process (</ table\_options> </ options>);

</ subordinate statements>;

RUN;

#### 4.1.37.3. Options

Option	Description
NODUPKEY	Eliminates duplicates in BY column values. As soon as two values in one column are identical, one of the lines is eliminated
OUT= output_table	Identifies the output_table. If you don't specify an output_table, eFront Script overwrites the table_to_process

TOP= number_of_lines	Defines the maximum number_of_lines to be included in the output_table
----------------------	--

#### 4.1.37.4. Subordinate statements

Statement	Description
BY	Identifies the columns used to sort the data table. You can specify more than one column.

#### 4.1.37.5. Comment

- If you sort more than one columns, the sort starts with the column next to BY.
- To keep the table\_to\_process intact, use the OUT statement, and define an output\_table.
- To prevent errors when **sorting strings**, use the BY together with the UPCASE option. Upcase changes temporarily all lowercase characters of the strings into uppercase characters. The sort operates only after the change.

#### 4.1.37.6. Output

No direct output.

#### 4.1.37.7. Error messages

Message	Explanation
ERROR(0113): {Column name} expected	No variable name has been specified
ERROR(0127): TOP value expected	The value assigned to the TOP option is incorrect.
Unknown BY column	The variable name that follows BY is invalid
Syntax error in data options definitions	Maybe you need to verify the usage conditions of the operators

### 4.1.38. PROC SQLLEXEC step

#### 4.1.38.1. Description

Independent program block. Always begins with a PROC SQLLEXEC statement and ends with a RUN statement. Launches the execution of an SQL query stored on a server.

#### 4.1.38.2. Syntax

PROC SQLLEXEC </ options>;

RUN;

#### 4.1.38.3. Options

Option	Description
CONNECTION= "named_connection"	<p>named_connection: name of the database connection that allows the system to access the target SQL database.</p> <p>Named connections must be defined beforehand (for details refer to the eFront Report Administrator Guide, <a href="#">Managing systematic access to external databases</a>).</p>
SQL= "EXEC query"	<p>Example:</p> <p>"EXEC dbo.sp_del_temp_report_date"</p> <p>(sp_del_temp_report_date is the query that is being stored on the server)</p>

#### 4.1.38.4. Comment

- Be very careful when using named connections. Errors may corrupt the database.

#### 4.1.38.5. Output

No direct output

#### 4.1.38.6. Error messages

Message	Explanation

### 4.1.39. PROC SQLIMPORT step

#### 4.1.39.1. Description

Independent program block. Always begins with the PROC SQLIMPORT statement and ends with a RUN statement. Imports SQL data into an eFront Report table. The SQL data may be from the database that is connected to the application, or from another database. Works with Microsoft SQL Server.

#### 4.1.39.2. Syntax

PROC SQLIMPORTDATA = destination\_table (</ table\_options> </ options>);

RUN:

### 4.1.39.3. Options

Option	Description
CONNECTION= "named_connection"	<p>You only need this option if you want to access a database that is not the one associated with the application!</p> <p>named_connection: name of the database connection that allows the system to access the target SQL database; defined in the file efront.config</p> <p>Named connections must be defined previously (for more information, refer to the eFront Report Administrator Guide, <a href="#">Managing systematic access to external databases</a>).</p> <p>(optional)</p>
TABLE=<SQL_table_to_import> SQL=<SQL_statement_to_execute> VIEW=<FrontReport_view_to_import>	<p>Any of the left-hand options allow you to define the set of data to be imported from the underlying database; mandatory option!</p> <p>You can do one of the following:</p> <ul style="list-style-type: none"> <li>refer to the name of a data table and use the TABLE option,</li> <li>directly use an SQL data query (SQL option),</li> <li>work with an eFront Report view (VIEW option).</li> </ul>
FILTER	<p>Can be used together with the 'TABLE' option.</p> <p>Specifies that user region filtering applies, but only if SOURCE = TABLE. The SQL case ?? FILTER is always correctly interpreted and gives the same result!</p> <p>(optional)</p>
NODATA	<p>Creates an eFront Report data bale, but void of data.</p> <p>(optional)</p>
MAXROWS= <integer>	<p>Specifies the maximum number of rows to be imported; Counting starts at the first table row.</p> <p>(optional)</p>

MAPPINGTABLE=<mapping_table_name>	<p>Reads the mapping from a mapping table in the FIA database and creates an eFront Script table with the following tables:</p> <ul style="list-style-type: none"> <li>• COLDBNAME - Database column name</li> <li>• COLFSNAME - eFront Script table column name</li> <li>• COLTYPE - Column type</li> <li>• COLCAPTION - Column caption</li> </ul> <p>(optional)</p>
COLUMNMAPPING=table_name	<p>The name of the table that contains the column names to be mapped.</p> <p>Must be followed by the COLUMNMAPPINGDBNAME and COLUMNMAPPINGFSNAME options.</p> <p>(optional)</p> <p> Not available for eFront Cube programs</p>
COLUMNMAPPINGDBNAME=database_column_name	<p>The name of the database column that is to be mapped to the column defined in the COLUMNMAPPINGFSNAME option.</p> <p>(optional)</p> <p> Not available for eFront Cube programs</p>
COLUMNMAPPINGFSNAME=eFront Report_column_name	<p>The name of the eFront Report column that is to be mapped to the column defined in the COLUMNMAPPINGDBNAME option.</p> <p>(optional)</p> <p> Not available for eFront Cube programs</p>

#### 4.1.39.4. Comment

- Be very careful when using named connections, and copy the exact labels!
- When using the COLUMNMAPPING option, the columns that are not in the mapping table retain their original names.
- When importing a list of rows and filters from the list of IQIDs, note the difference in the SQL format between the FrontReport and FrontCube PROC SQLIMPORT.

As shown in the examples below, FrontCube P\_SQLIMPORT is using a single SQL parameter of the type list, e.g. SQL = "SELECT \* FROM VCFUNDOP WHERE FUND IN (SELECT IQID FROM ?);{FUNDS}";.

Sample FrontReport program:

```
PROC SQLIMPORT DATA=WORK.MF SQL = "SELECT IQID FROM VCFUND";
RUN; %LET FUNDS = COLLECTION("WORK.MF","IQID"); PROC SQLIMPORT
DATA=WORK.MF_OPERATIONS SQL = "SELECT * FROM VCFUNDOP WHERE
FUND IN (?)";{FUNDS}; RUN;
```

Sample FrontCube program:

```
PROC SQLIMPORT DATA=WORK.MF SQL = "SELECT IQID FROM VCFUND";
RUN; %LET FUNDS = COLLECTION("WORK.MF","IQID"); PROC SQLIMPORT
DATA=WORK.MF_OPERATIONS SQL = "SELECT * FROM VCFUNDOP WHERE
FUND IN (SELECT IQID FROM ?);{FUNDS}"; RUN;
```

#### 4.1.39.5. Output

Data imported in an eFront Report data table.

#### 4.1.39.6. Error messages

Message	Explanation
---------	-------------

### 4.1.40. PROC SQLTABLE step

#### 4.1.40.1. Description

Independent program block. Always begins with the PROC SQLTABLE statement and ends with a RUN statement. Exports an eFront Report table to a SQL database.

#### 4.1.40.2. Syntax

```
PROC SQLTABLEDATA = table_to_export (</ table_options> </ options>);
```

RUN;

#### 4.1.40.3. Options

Option	Description
--------	-------------

CONNECTION= "named_connection"	named_connection: name of the database connection that allows the system to access the target SQL database.  Named connections must be defined previously (for details refer to the eFront Report Administrator Guide, <a href="#">Managing systematic access to external databases</a> ).  mandatory;
GROUPBY = <integer>	
CREATE	Flag that indicates that a table must be created.  mandatory;
NOERROR	By default, if the table to be created exists already, and error is generated, the procedure comes to an halt. But if the flag NOERROR is included, the programming step continues to execute.  optional;
TABLE= "table_name"	Name of the SQL table to be created in the SQL database.  mandatory;
TEXT=column_name	Selects the columns to be kept in the table. Use a blank space to separate column names.  This option indicates columns that contain text data longer than 254 characters!  optional;
MAPPINGTABLE="mapping_table_name"	When used, automatically creates a mapping between the columns in the eFront Script program table and the database. The mapping table contains the following columns: <ul style="list-style-type: none"><li>• Database column name</li><li>• eFront Script table column name</li><li>• Column type</li><li>• Column caption</li></ul> optional;

COLUMNMAPPING=table_name	The name of the table that contains the column names to be mapped.  Must be followed by the COLUMNMAPPINGDBNAME and COLUMNMAPPINGFSNAME options.  (optional)   Not available for eFront Cube programs
COLUMNMAPPINGDBNAME=database_column_name	The name of the database column that is to be mapped to the column defined in the COLUMNMAPPINGFSNAME option.  (optional)   Not available for eFront Cube programs
COLUMNMAPPINGFSNAME=eFront Report_column_name	The name of the eFront Report column that is to be mapped to the column defined in the COLUMNMAPPINGDBNAME option.  (optional)   Not available for eFront Cube programs

#### 4.1.40.4. Comment

- Be very careful when using named connections. Errors may corrupt the database.

#### 4.1.40.5. Output

Table in a SQL database.

#### 4.1.40.6. Error messages

Message	Explanation

### 4.1.41. PROC TABULATE step

#### 4.1.41.1. Description

Independent program block. Always begins with the PROC TABULATE statement and ends with a RUN statement. Creates a **pivot table**. A pivot table is a **data**

**summarization tool** which calculates statistics for the data stored in the input data table and creates a second table with the calculated data.

 Not available for eFront Cube programs.

#### 4.1.41.2. Syntax

```
PROC TABULATE DATA = table_to_process (</ table_options> </ options>);

</ subordinate statements>

RUN;
```

#### 4.1.41.3. Options

Option	Description
<b>FORMAT</b> =	Specifies the format to apply to all columns. Only useful if all columns are of the same type.
<b>LABEL</b>	Specifies that column labels display as column titles.
<b>STYLE</b> =stylesheet_name	Applies an existing stylesheet to PROC TABULATE items. The stylesheet definition must precede the PROC TABULATE step.
<b>STYLE</b> (item_to_be_styled) = (table_style_attributes)	Applies a particular style to a PROC TABULATE item.

#### 4.1.41.4. Subordinate statements

Statement	Description
<b>BOX</b>	Specifies the text to place in the empty box above row titles
<b>CLASS</b>	Identifies the data category of analysis. eFront Script calculates statistical values for each value of the CLASS columns.
<b>FORMAT</b>	Assigns a format to one or more columns
<b>TABLE</b>	Specifies the organization and the appearance of the output table
<b>TITLE</b>	Specifies the table title. 10 hierarchical levels are available
<b>VAR</b>	Identifies the variables on which to perform the statistical analysis. Missing values are only taken into account, if values are counted.

#### 4.1.41.5. Comment

- The PROC TABULATE step creates direct output.
- The input table is not modified!
- The TABLE and Class statements are required.

#### 4.1.41.6. Output

HTML output.

#### 4.1.41.7. Error messages

Message	Explanation
---------	-------------

### 4.1.42. PROC TRANSPOSE step

#### 4.1.42.1. Description

Independent program block. Always begins with the PROC TRANSPOSE statement and ends with a RUN statement. Creates an output table by restructuring the input table, transposing selected columns into lines, and creating new columns. PROC TRANSPOSE can often eliminate the need to write a lengthy DATA step to achieve the same result. Further, the output data set can be used in subsequent DATA or PROC steps for analysis, reporting, or further data manipulation.

#### 4.1.42.2. Syntax

```
PROC TRANSPOSE DATA = table_to_process (</ table_options> </ options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

#### 4.1.42.3. Output Options

Option	Description
OUT= output_table	Identifies the output_table
NAME= col_name	Specifies the name of the output table column that contains the name of the variables being transposed

LABEL="label"	Specifies the label of the output table column, that contains the labels of the variables being transposed
PREFIX=prefix	Specifies a prefix to use in constructing names for transposed variables in the output data set. The default value for PREFIX= COL. The default names for transposed variables are: "COL1" "COL2"..."COLn". For more details, refer to <a href="#">Naming transposed variables</a> . prefix: no quotation marks!

#### 4.1.42.4. Subordinate statements

Statement	Description
<code>VAR</code> col_1col_n;	Identifies the input table columns to be transposed
<code>ID</code> col	Identifies the input table column whose formatted values name the variables being transposed.
<code>IDLABEL</code> col	Specifies the column labels for the variables being transposed (labels display when using the PROC PRINT step with the LABEL option)
<code>ID TYPE</code>	Identifies the input table type
<code>ID FORMAT</code>	

#### 4.1.42.5. Comment

- As input table, you can only use tables which have different values of the column to be used as key.
- The number of columns of the output table = the number of lines of the input table + 1.
- By default, **eFront Script** generates automatically column names.
- If you don't specify an output table, **eFront Script** overwrites the input table.
- When using PREFIX= with an ID statement, the value prefixes to the ID value.
- By default, the columns being transposed are numeric columns. If you want to force the column transposition into a text format for example, you need to add allcolumns.

#### 4.1.42.6. Output

No printed output.

#### 4.1.42.7. Error messages

Message	Explanation
<code>^ ERROR(0113): {prefix} expected</code>	Either the PREFIX value is missing, or it is invalid. If enclosed in quotation strings, the PREFIX value is invalid!

## 4.1.43. PROC TRANSPOSE2 step

### 4.1.43.1. Description

Independent program block. Always begins with the PROC TRANSPOSE2 statement and ends with a RUN statement. Creates an output table by restructuring the input table: the values of a selected column are used to create additional table columns, and the values of another selected column in the initial table are written into the new columns.

### 4.1.43.2. Syntax

```
PROC TRANSPOSE2 DATA = table_to_process (</ table_options> </ options>);  
</ subordinate statements>;  
RUN;
```

### 4.1.43.3. Options

Option	Description
OUT= output_table	Identifies the output_table.
LABEL= col_name	Refers to the column of the table_to_process whose values will be used as table columns.
VALUE= col_name	Refers to the column of the table_to_process whose values will be displayed in the columns created from the column referred to under 'Label'.

### 4.1.43.4. Subordinate statements

Statement	Description
CLASScol_acol_x;	Identifies the columns of the table_to_process to be used as key for grouping lines.
VARcol_1col_n;	Identifies the columns of the table_to_process to be kept in the output_table.

### 4.1.43.5. Comment

- As input table, you will use tables which have several identical values of the column to be used as key.

### 4.1.43.6. Output

No printed output.

#### 4.1.43.7. Error messages

Message	Explanation
<code>^ ERROR(0113): {prefix} expected</code>	Either the PREFIX value is missing, or it is invalid. If enclosed in quotation strings, the PREFIX value is invalid!

### 4.1.44. PROC UPDATE step

#### 4.1.44.1. Description

Independent program block. Always begins with the PROC UPDATE statement and ends with the RUN statement. Merges two eFront Cube cubes (left join merge).

 Applies only to eFront Cube programs.

#### 4.1.44.2. Syntax

PROC UPDATE </ options>;

</ subordinate statements>;

RUN;

#### 4.1.44.3. Options

Option	Description
DATA=cube	<p>cube: name of cube 1</p> <p>Example:</p> <pre>DATA = SESSION.T_Dashboard_Main_Table</pre> <p>mandatory;</p>
WITH=cube	<p>cube: name of cube 2</p> <p>Example:</p> <pre>WITH = WORK.T_FUND_BLOB_ADVISORYBOARDTABLE_Last</pre> <p>mandatory;</p>

CLASScolumn_name	column_name: defines the column to be used as merge criteria  Example:  CLASS Fund_Id  mandatory;
------------------	---

#### 4.1.44.4. Subordinate statements

Option	Description
CLASScolumn_name	column_name: defines the column to be used as merge criteria  Example:  CLASS Fund_Id  mandatory;

#### 4.1.44.5. Comment

- PROC UPDATE requires at least two input cubes.
- The merge doesn't impact the number of cube rows by adding or removing rows.

#### 4.1.44.6. Output

Merged eFront Cube cube.

#### 4.1.44.7. PROC UPDATE and DATA MERGE - Comparison

The PROC UPDATE step is very similar to the DATA MERGE statement with the limitation that it is impossible to add or remove rows. Compared to DATA MERGE, PROC UPDATE has much better performance as the impacted columns are not recalculated. In addition, PROC UPDATE and DATA MERGE differ in the way the cubes are merged. For example, if 1 row in the first cube matches N rows in the right cube:

- when using DATA MERGE, the system outputs N rows in the merged cube
- when using PROC UPDATE, the system outputs 1 row in the merged cube

#### 4.1.44.8. Error messages

Message	Explanation
---------	-------------


## 4.1.45. PROC WSGETCOMPANYDATA

### 4.1.45.1. Description

Independent program block. Always begins with the PROC WSGETCOMPANYDATA statement and ends with a RUN statement. Retrieves company data from eFront PEO/VC for use in eFront Portfolio Monitoring.

 Not available for eFront Cube programs.

### 4.1.45.2. Syntax

PROCWSGETCOMPANYDATA </ options>;

RUN;

### 4.1.45.3. Options

Option	Description
COMPANYID = company_IQID	The eFront Invest IQID of the company. mandatory;
OUT = output_table	The eFront Script table that will store the company data. mandatory;

### 4.1.45.4. Comment

- eFront PEO/VC and eFront Portfolio Monitoring must be connected through the FrontPM Web Service in order to use PROC WSGETCOMPANYDATA.
- The company data must be published in eFront PEO/VC before it can be retrieved using PROC WSGETCOMPANYDATA.

### 4.1.45.5. Output

An eFront Script table with the following columns:

- UNIQUE\_ID

- COMPANY\_NAME
- STATUS
- INDUSTRY
- COMPANY\_CURRENCY
- NATURE\_OF\_BUSINESS
- GEOGRAPHY
- COUNTRY

#### 4.1.45.6. Error messages

Message	Explanation
---------	-------------

### 4.1.46. PROC WSGETINVESTMENTDATA

#### 4.1.46.1. Description

Independent program block. Always begins with the PROC WSGETINVESTMENTDATA statement and ends with a RUN statement. Retrieves company investment data from eFront PEO/VC for use in eFront Portfolio Monitoring.

 Not available for eFront Cube programs.

#### 4.1.46.2. Syntax

PROCWSGETCOMPANYDATA </ options>;

RUN;

#### 4.1.46.3. Options

Option	Description
INVESTMENTID = investmentid	The eFront Invest InvestmentID. mandatory;
OUT = output_table	The eFront Script table that will store the investment data. mandatory;

#### 4.1.46.4. Comment

- eFront PEO/VC and eFront Portfolio Monitoring must be connected through the FrontPM Web Service in order to use PROC WSGETINVESTMENTDATA.
- The investment data must be published in eFront PEO/VC before it can be retrieved using PROC WSGETINVESTMENTDATA.

#### 4.1.46.5. Output

An eFront Script table with the following columns:

- UNIQUE\_ID
- ND\_OWNERSHIP
- FD\_OWNERSHIP
- INVESTMENTS
- IRR
- MULTIPLE

#### 4.1.46.6. Error messages

Message	Explanation
---------	-------------

### 4.1.47. PROC WSGETINVESTORS DATA

#### 4.1.47.1. Description

Independent program block. Always begins with the PROC WSGETINVESTORS DATA statement and ends with a RUN statement. Retrieves company investor data from eFront PEO/VC for use in eFront Portfolio Monitoring.

 Not available for eFront Cube programs.

#### 4.1.47.2. Syntax

PROCWSGETINVESTORS DATA </ options>;

RUN;

### 4.1.47.3. Options

Option	Description
COMPANYID = company_IQID	The eFront Invest IQID of the company. mandatory;
OUT = output_table	Specifies the eFront Script table that will store the company data. mandatory;

### 4.1.47.4. Comment

- eFront PEO/VC and eFront Portfolio Monitoring must be connected through the FrontPM Web Service in order to use PROC WSGETINVESTORSDATA.
- The investor data must be published in eFront PEO/VC before it can be retrieved using PROC WSGETINVESTORSDATA.

### 4.1.47.5. Output

An eFront Script table with the following columns:

- INVESTOR
- SECURITY
- CURRENT\_CASH

### 4.1.47.6. Error messages

Message	Explanation

## 4.1.48. STYLESHEET step

### 4.1.48.1. Description

Independent program block. Always begins with a STYLESHEET statement and ends with a STYLESHEET END statement. The STYLESHEET step must precede the programming steps it applies to. Stylesheet definitions can be applied to: PROC PRINT, PROC PRINTCOL, PROC TABULATE, PROC GCHART steps. A stylesheet regroups different styles applicable to the elements used by the output formatting steps.

## 4.1.48.2. Syntax

```
STYLESHEETstylesheet_name;
step_name (item_to_be_styled_1) = (</ style_attribute = value >);
step_name (item_to_be_styled_n) = (</ style_attribute=value >);
STYLESHEET END;
```

## 4.1.48.3. Style attributes

There are 2 types of style attributes:

- Text style attributes and values
- Chart style attributes and values

You choose style attributes and values according to step and step elements to be styled:

Step	Items to be styled	
PRINT	BANNERFILE	N
	BANNER	OBS
	COLHEADER	TITLE
	DATA	TITLE1
	GRANDTOTAL	TOTAL
	HEADER	
PRINTCOL	DATA	
	TITLE	
	TITLE1	
TABULATE	BANNERFILE	DATA
	BANNER	HEADER
	BOX	TITLE
	CLASS	TITLE1

GCHART	DONUT DONUT3D HBAR HBAR3D PIE	PIE3D TITLE TITLE1 VBAR VBAR3D
--------	---	--

#### 4.1.48.4. Comment

- By default the STYLESHEET statement applies to all programming steps it precedes.
- stylesheet\_name is only required, if the stylesheet applied by default is not appropriate.
- To apply a stylesheet to a programming step, use the **STYLE** option.
- To re-set default styles, use:  
**STYLESHEET;**  
**STYLESHEET END;**

#### 4.1.48.5. Output

No output

#### 4.1.48.6. Error messages

Message	Explanation
Stylesheet not found	The stylesheet cannot be found
ERROR: Invalid style element	The item to be styled may be invalid.
Cannot find image :	The image being referred to by PRINT(BANNERFILE) cannot be found
ERROR: Invalid style parameter	The style attribute being used is not correct or the item to be styled is not valid

### 4.1.49. TABLE step

#### 4.1.49.1. Description

Always begins with the TABLE statement and ends with an END statement. Allows you to dispatch printed output in an HTML table. The options to be used together with

the TABLE, and ROW statements correspond to the current HTML attributes. In the following we don't list all the possible attributes, we only show an example of usage.

#### 4.1.49.2. EXAMPLE

```
TABLE (COL =("50%" "50%")STYLE = (WIDTH ="100%" PADDING ="5px"));
```

```
ROW (STYLE = (VALIGN ="TOP" PADDING ="x%"))
```

```
CELL;
```

```
PROC PRINT | PROC PRINTFORM
```

```
CELL END;
```

```
CELL;
```

```
PROC PRINT | PROC PRINTFORM
```

```
CELL END;
```

```
ROW END;
```

```
...
```

```
END;
```

#### 4.1.49.3. Output

Direct output.

#### 4.1.49.4. Error messages

Message	Explanation
---------	-------------

## 4.2. Embeddable steps

Embeddable steps can occur within independent programming steps. Embeddable steps are program **blocks** that start with an opening statement and end with a closing statement. Each step is composed of a series of **statements** that allow, according to the step's overall functionality, to specify the actions to undertake.

Embeddable steps are context sensitive, i.e. occur only within a specific environment (main programming step).

Main step	Embeddable step	Main purpose
PROC PRINT	EVENT	Embed a control structure within the main step. Useful when applying conditional cell coloring to the printed table, for example.
PROC PRINTCOL		
PROC PRINTFORM		

### 4.2.1. EVENT step

#### 4.2.1.1. Description

Embeddable step; valid in: PROC PRINT steps. Always begins with the EVENT statement and ends with an END statement. Enables the embedding of a control structure within the PROC PRINT step.

#### 4.2.1.2. Syntax

PROC PRINT DATA =table\_to\_process (</ table\_options>);

EVENT PREDATAROW

IF <condition> THEN

</ subordinate statements>;

ELSEIF <condition> THEN

</ subordinate statements>;

ELSE

```
</ subordinate statements>;  
END;</ subordinate statements>;  
END;  
RUN;
```

#### 4.2.1.3. Options

Option	Description
PREDATAROW	Required;

#### 4.2.1.4. Comment

- EVENT always needs a keyword right next to it.
- Can be used for conditional cell coloring, for example.

#### 4.2.1.5. Output

Direct output

#### 4.2.1.6. Error messages

Message	Explanation
---------	-------------

## 4.3. Global statements

**Global** statements are global in scope and apply to any programming step they precede. **eFront Script** handles only one global statement:

- **LIBNAME** statement

### 4.3.1. LIBNAME statement

#### 4.3.1.1. Description

LIBNAME is a [global statement](#). Assigns an alias to a library (folder). From within a program, table names can refer to the alias, if prefixed by it. The LIBNAME statement must precede all programs that read from or write to the library.

By default, **eFront Script** uses the WORK (system) option. WORK is a temporary library created at the beginning of each program execution in users' sessions in their virtual memory space. Temporary WORK libraries are deleted at the end of each program execution. To manage memory usage of WORK libraries, specific arguments are available!

#### 4.3.1.2. Syntax

```
LIBNAMElibrary_alias "path_to_library" </work_lib_arguments>;
```

```
DATAlibrary_alias.table_to_process (</ table_options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

#### 4.3.1.3. WORK Library Arguments

Attribute	Description
<b>DISK</b>	Argument, that applies only to 'WORK' libraries. Allows you to reduce memory usage and avoid Out-Of-Memory exceptions.  LIBNAME WORK DISK specifies the disk storage of tables instead of in-memory storage. Each reference to temporary 'WORK' libraries will be executed in disk storage mode.

MEMORY	<p>Argument, that applies only to 'WORK' libraries.</p> <p>LIBNAME WORK MEMORY restores the use of the in-memory storage of tables in temporary 'WORK' libraries.</p>
--------	---

#### 4.3.1.4. Comment

- The library\_alias must refer to an already existing folder. You cannot create new folders using the LIBNAME statement.
- Use LIBNAME USER CURRENT to refer to the folder currently opened by the user.
- If you change the language of your interface once programs have been written, be aware that **eFront Report** automatically modifies folder names (ex.: **Privé** is translated in **Private**). As a consequence, programs may no longer execute correctly, especially when using absolute paths in the LIBNAME statement. The same happens if you change folder names. To work around this difficulty, apply two methods: use **magic variables**. Use the LIBNAME statement as a program to include. If LIBNAME statements are set up in a program to include, changes only have to be made within this program.

#### 4.3.1.5. Error messages

Message	Explanation
<a href="#">Invalid or unknown library</a>	The library cannot be identified.
<a href="#">Invalid path or path not found</a>	The library (folder) referred to does not exist or cannot be found.

#### 4.3.1.6. DISK argument

##### Description

Argument; used to specify the behavior of WORK libraries defined/used in the [LIBNAME statement](#) and the [DATA step](#).

- If used together with a LIBNAME statement, DISK specifies the disk storage of tables instead of in-memory storage. Each reference to temporary WORK libraries will be executed in disk storage mode.
- If used together with a DATA step: you can replace the WORK library prefix by DISK, meaning that WORK tables are stored on disk instead of in memory.

## Syntax (LIBNAME statement)

```
LIBNAME WORK DISK;
```

## Syntax (DATA step)

```
DATA DISK.output_table (</ table_options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

## Comment

- The principle is to store WORK tables on disk instead of in memory, and to read them line by line when needed. Therefore, these tables are never completely load into memory..

## Example Syntax (DATA step)

Replace:

```
DATA WORK.My_table;
```

```
SET LIBRARY VIEW.My_view;
```

```
RUN;
```

By:

```
DATA DISK.My_table;
```

```
SET LIBRARY VIEW.My_view;
```

```
RUN;
```

### 4.3.1.7. MEMORY argument

#### Description

Argument; used to specify the behavior of WORK libraries defined/used in the [LIBNAME statement](#) and the [DATA step](#).

- If used together with a LIBNAME statement, MEMORY restores the use of the in-memory storage of tables in temporary WORK libraries.

### Syntax (LIBNAME statement)

LIBNAME WORK MEMORY;

### Comment

- As long as the storage in memory is not restored, the storage on disk applies after the statement LIBNAME WORK DISK.

## 4.4. Macro statements

**Macro** statements allow to customize **eFront Script** programs and simplify programming steps. Macro statements are global in scope, apply to any programming step they precede, and undergo **pre-processing**. A macro statement instructs the macro processor to perform an operation. **Macro statements** are prefixed by a %. Macro statements are not allowed within DATA or PROC steps. Macro statements can create macro variables, that can be referenced from within DATA or PROC steps. **Macro variables** are prefixed by a %.

Here is the list of available macro statements:

- %**DEFINE** statement
- %**CONTINUE** statement
- %**INCLUDE** statement
- %**LET** statement
- %**PARAM** statement

### 4.4.1. %**DEFINE** statement

#### 4.4.1.1. Description

The %**DEFINE** statement is a **macro statement**. Use this statement to declare the **existence of a macro**.

#### 4.4.1.2. Syntax

```
%DEFINEmacro_name;  
step_nameDATA = table_to_process (</ table_options>);  
</ subordinate statements>;  
RUN;
```

#### 4.4.1.3. Comment

- You might want to declare the existence of a macro, if your **eFront Script** program behaves differently according to the existence/non-existence of a macro. In this case,

you would start out, declaring a macro, and then test the existence of the macro using the EXISTMACRO() function.

## 4.4.2. %INCLUDE statement

### 4.4.2.1. Description

The %INCLUDE statement is a [macro statement](#). Includes eFront Script programming steps, statements, or data lines into a current eFront Script program.

### 4.4.2.2. Syntax

```
%INCLUDE "program_name" </ options>;  
  
step_nameDATA = table_to_process (</ table_options>);  
  
</ subordinate statements>;  
  
RUN;
```

### 4.4.2.3. Options

Option	Description
NORELOAD	Avoids reloading files that have already been loaded.

### 4.4.2.4. Comment

- The %INCLUDE statement precedes programming steps.
- If the host program and the program to include, share the same location, "program\_name" = name of the program to include.
- If the host program and the program to include, do **not** share the same location, "program\_name" = [access path](#) + name of the program to include.
- To make your program calls more robust, and avoid compilation errors due to changing directory names, use [magic variables](#) to define the access path to programs.

### 4.4.2.5. Error messages

Message	Explanation
---------	-------------

Include object not found	The object to be included couldn't be located. It is likely that the <a href="#">path specification</a> is erroneous.
ERROR(0113): 'RUN' expected	RUN expected. This message is likely to appear if you place the %INCLUDE statement inside a program step.

### 4.4.3. %CONTINUE statement

#### 4.4.3.1. Description

Valid in: **macro controls**. Exits the current iteration of the control loop, and jumps to the following statement.

#### 4.4.3.2. Syntax

```
</ macro_control_start_statement>;
%CONTINUE;
</ macro_control_end_statement>;
```

#### 4.4.3.3. Comment

- The %CONTINUE statement is applicable to [%DO WHILE...%LOOP WHILE](#), [%FOR...%NEXT](#), and [%WHILE...%END](#) loops.

#### 4.4.3.4. Error messages

Message	Explanation

### 4.4.4. %LET statement

#### 4.4.4.1. Description

The %LET statement is a [macro statement](#). %LET defines a [macro-variable](#) and assigns a static value to it.

#### 4.4.4.2. Syntax

```
%LETmacro_var_name = "value";
DATAoutput_table (</ table_options>);
```

```
SET%macro_var_name;  
ColumnMyColumnType=Float;  
MyColumn = %macro_var_name;  
RUN;
```

#### 4.4.4.3. Comment

- The %LET statement precedes DATA and PROC steps.
- value: values are character strings.
- value: can result from a function being applied, such as:  
  %LET InstrLabel1 = LOOKUP("VCINVESTMENTS.USERTEXT16",1);  
  For details, refer to [Functions - special](#).
- Use the %LET statement to debug programs.
- When testing programs that require dynamic user input, replace the %LET statement by the %PARAM statement.

#### 4.4.5. %PARAM statement

##### 4.4.5.1. Description

The %PARAM statement is a [macro statement](#). %PARAM defines macro-variables that are instantiated by interactive user input.

##### 4.4.5.2. Syntax

```
%PARAMparam_name </options>;  
DATAoutput_table (</ table_options>);  
COLUMNcol_name;  
col_name = %param_name;  
RUN;
```

### 4.4.5.3. Options

Option	Description
LABEL= "<label>"	<label>: Specifies the text to display when asking for user input. The text that displays by default is the name of the parameter.  To adapt to bilingual user interfaces, specify the label as follows: LABEL = "<English_label>{F}<French_label>"
TYPE= <type>	Identifies the type of data required. <type> = DATE   STRING   INTEGER   FLOAT   BOOLEAN. If type = DATE, the user can select a date from a pop-up calendar.  If TYPE = STRING, it is not possible to change the size of the text box that displays automatically.
NOTNULL	Forces the user to enter a value.
DEFAULT= <value>  DEFAULT= "<value>"	Identifies a default value to display for interactive user input choice.  <value>: dynamic value being calculated using a function.  <value>: static character string  Example: TYPE= DATE DEFAULT=TODAY  Example: TYPE= DATE DEFAULT="01012006"d (refer to <a href="#">Standard eFront Script formats</a> )  Example: TYPE= STRING DEFAULT="Mary"
INFORMAT="<informat>(<data_container>)"	Identifies a list of default values to display for interactive user input choice. The user can choose one or more values. According to the <informat>, the <data_container> varies, as does the type of values returned. Refer to the <a href="#">list of available informats</a> .
SHOWSELECTEDITEMS	Shows in the user interface the items being selected by the user.  You must use SHOWSELECTEDITEMS in combination with multi-items selection informats, such as ??XID and ??XPICKID
NBROWS= <Numeric>	<Numeric>: Specifies the number of rows displayed for a defined list of values.

#### 4.4.5.4. Comment

- The %PARAM statement precedes DATA and PROC steps.
- You can point from within an SQL query to user active input using the %PARAM statement.
- To optimize program testing, comment the %PARAM statement out, and use the %LET statement during testing.
- You can use several %PARAM statements within a program.
- You can link two %PARAM statements, and use the value(s) returned by the first one to define the data selection to be presented to the user by the second one.

#### 4.4.5.5. Error messages

Message	Explanation
---------	-------------

#### 4.4.6. %Param - Informats

There are two types of **informats**:

- Informats that return a **single value**
- Informats that return a **list of values** (informat keywords start with an "X").

Informats that return a list of values cannot be used:

- Together with IN expressions.
- Together with a PUT statement in a PROC PRINT step.

You cannot use datamart tables together with informat definitions!

Always use Capital letters! If you don't, you may experience issues when moving to eFront Invest.

Option	Description	Example
INFORMAT= "?ID(<data_table>)"	Returns 1 ID corresponding to the value being chosen in the <data_table>	INFORMAT="??ID(VCFUND)" Displays all the funds contained in the VCFUND the ID of the fund being chosen.

<code>INFORMAT= "??XID(&lt;data_table&gt;)"</code>	Returns an ID list corresponding of the values being chosen in the <data_table>	
<code>INFORMAT= "??CHOICE(&lt;value_1&gt;, &lt;value_n&gt;)"</code>	Displays a scrolling list, and returns the <value> being chosen.	<code>INFORMAT="??CHOICE(AX,OF,BF,RV,PVI)"</code> Displays a scrolling list composed of 5 elements value being chosen. You can use the ??CHOICE option to create a list based on year instead of date: <code>INFORMAT="??CHOICE(2006, 2007, 2008, 2009, 2010)"</code>
<code>INFORMAT= "??CHOICE(&lt;id_1&gt;=&lt;value_1&gt;, &lt;id_n&gt;=&lt;value_n&gt;)"</code>	Displays a scrolling list, and returns the <id> corresponding to the value being chosen.	<code>%PARAM CHOICE LABEL="Test Choice" INFORMAT="??CHOICE(VAL1=Valeur 1,VAL2=Valeur 2);"</code>
<code>INFORMAT= "??XCHOICE(&lt;id_1&gt;=&lt;value_1&gt;, &lt;id_n&gt;=&lt;value_n&gt;)"</code>	Displays a list with checkboxes and allows users to choose several values.	<code>%Param XCHOICE label="Test XChoice" informFORMAT="??XCHOICE(VAL1=Valeur 1,VAL2=Valeur 2);"</code> You can also use the DEFAULT option to define checked by default: <code>%PARAM XCHOICE</code> <code>LABEL = "Fruit"</code> <code>INFORMAT = "??XCHOICE(1=Apple,2=Orange,3=Pear,4=Strawberry,5=Banana,DEFAULT="1,3";"</code>
<code>INFORMAT= "??CHECK(&lt;selection_1&gt;, &lt;selection_n&gt;, All)"</code>	Returns the position of the values being chosen.	
<code>INFORMAT= "??PICKID(&lt;data_table&gt;; &lt;col_name&gt;)"</code>	Displays a scrolling list of all the <col_name> values, and returns the ID of the value being chosen.	<code>INFORMAT="??PICKID(VCFUND;FUND);"</code> Displays all the funds, and returns the ID of the fund.

<code>INFORMAT= "??PICKID(&lt;data_table&gt;;&lt;col_name&gt;;&lt;expression&gt;)"</code>	Displays a scrolling list of all the <code>&lt;col_name&gt;</code> values, and returns the ID of the value being chosen.	<code>INFORMAT="??PICKID(VCFUND;FUND;FSTA"</code> Displays all the funds with the FStatus set to zero ID of the fund being chosen.
<code>INFORMAT= "??PICKID(&lt;SQL query&gt;)"</code>	Displays a scrolling list of all the values returned by the SQL query, and returns the ID of the value being chosen.	
<code>INFORMAT= "??PICKID(&lt;SQL query WHERE (?? FILTER)&gt;)"</code>	<code>??FILTER</code> used together with the <code>&lt;SQL-query&gt;</code> passed to the <code>?? PICKID</code> keyword allows to filter the <code>&lt;data_table&gt;</code> according to the user region.	Refer to example: <a href="#">%PARAM - ??PICKID(SQL- query) - ??FILTER</a>
<code>INFORMAT= "??XPICKID(&lt;data_table&gt;; &lt;col_name&gt;)"</code>	Displays a scrolling list of all the <code>&lt;col_name&gt;</code> values, and returns the list of IDs of the values being chosen.	
<code>INFORMAT= "??XPICKID(&lt;SQL-query&gt;)"</code>	Displays a list of all the values returned by the SQL query, and returns the list of ID's of the values being chosen.	

<pre>INFORMAT= "??XPICKID(&lt;SQL-query WHERE (?? FILTER)&gt;)"  or  INFORMAT= "??XPICKID(&lt;SQL-query WHERE {FILTER}&gt;)"</pre>	<p>??FILTER used together with the &lt;SQL-query&gt; passed to the ?? XPICKID keyword allows to filter the &lt;data_table&gt; according to the user region.</p> <p>You cannot apply default values to this parameter!</p>	<p>Refer to example:</p> <p><a href="#">%PARAM - ??XPICKID with SQL query</a></p>
<pre>INFORMAT= "??XPICKID(&lt;SQL-query (?)&gt;,{&lt;var&gt;})"  INFORMAT= "??XPICKID(&lt;SQL-query (?)({?})&gt;,{&lt;var&gt;}; &lt;var&gt;)"</pre>	<p>You can link two %PARAM statements, and use the value(s) returned by the first one to define the data selection to be presented to the user by the second one.</p> <p>This is only possible with ?? XPICKID!</p>	<p>Refer to examples:</p> <p><a href="#">%PARAM - ??PICKID_{VAR}</a></p> <p><a href="#">%PARAM - ??XPICKID - (?) - (?)</a></p>

<p>INFORMAT= "??LOOKUPCODE(&lt;reference_table&gt;)"</p> <p>or</p> <p>INFORMAT= "?? LOOKUPCODE(&lt;reference_table&gt;.&lt;additional_field&gt;)"</p>	<p>Applies to standard reference tables, and user defined reference tables. Reference tables contain the list of options that are associated with standard fields. User defined reference tables contain the list of options that are associated with additional fields.</p> <p>?? LOOKUPCODE displays a scrolling list of the options contained in the &lt;reference_table&gt; and returns the code of the option selected.</p>	<p>INFORMAT="??LOOKUPCODE(VCINVESTINSCLAS")</p> <p>Returns the code being associated to one of the reference table VCINVESTINSCLAS</p>
---	--	--

<p>INFORMAT= "??LOOKUP(&lt;reference_table&gt;)" or INFORMAT= "?? LOOKUP(&lt;reference_table&gt;.&lt;additional_field&gt;)"</p>	<p>Applies to standard reference tables, and user defined reference tables. Reference tables contain the list of options that are associated with standard fields. User defined reference tables contain the list of options that are associated with additional fields.</p> <p>??LOOKUP displays a scrolling list of the options contained in the &lt;reference_table&gt; and returns the description (label) of the option selected.</p>	<p>Refer to exemple: <a href="#">%PARAM - ??LOOKUP - ??XLOOKUPCODE</a></p>
---	--	--

<p>INFORMAT= "??XLOOKUPCODE(&lt;reference_table&gt;)"</p> <p>or</p> <p>INFORMAT= "?? XLOOKUPCODE(&lt;reference_table&gt;.&lt;additional_field&gt;)"</p>	<p>Applies to standard reference tables, and user defined reference tables. Reference tables contain the list of options that are associated with standard fields. User defined reference tables contain the list of options that are associated with additional fields.</p> <p>?? XLOOKUPCODE displays the list of the options contained in the &lt;reference_table&gt; and returns the codes of the options selected.</p>	<p>Refer to exemple:</p> <p><a href="#">%PARAM - ??LOOKUP - ??XLOOKUPCODE</a></p>
---	---	---

<p>INFORMAT= "??XLOOKUP(&lt;reference_table&gt;)"</p> <p>or</p> <p>INFORMAT= "?? XLOOKUP(&lt;reference_table&gt;.&lt;additional_field&gt;)"</p>	<p>Applies to standard reference tables, and user defined reference tables. Reference tables contain the list of options that are associated with standard fields. User defined reference tables contain the list of options that are associated with additional fields.</p> <p>??XLOOKUP displays the list of the options contained in the &lt;reference_table&gt; and returns the descriptions (labels) of the options selected.</p> <p>You cannot apply default values to this parameter!</p>	<p>Refer to exemple:</p> <p><a href="#">%PARAM - ??LOOKUP - ??XLOOKUPCODE</a></p>
---	--	---

Informats can only be used together with **application owned data tables** (principal tables). The most current application owned tables are:

Table name	Description
VCACCTCAT	
VCBANKACCOUNT	returns the list of bank accounts being entered in the database
VCBENCHMARK	
VCFUND	returns the list of the funds being invested in, and investing funds
VCFUNDOP	returns the list of fund operations
VCFUNDOTHER	
VCFUNDSHARE	returns the list of fund shares
VCINVESTMENT	returns the global investment for an investor

VCINVESTMENTINS	returns the details of investments for an investor in an instrument; list of investment instruments
VCINVESTMENTTOP	returns the list of operations linked to investment instruments
VCINVESTCLASS	returns the list of security classes being entered in the database
VCINVESTOPTYPE	
VCINSTRCONV RATE	
VDFUNDINVPOLICY	returns text items that express the investment policy
VCNOTE	returns the list of all notes being entered in the database
VCPUBLICINFO	
VCPUBLICMARKET	
VCSHAREHOLDCAT	List of investor categories
VCSHAREHOLDTYPE	List of investor types
VCSUBSCRIBER	returns the list of fund subscribers
VCSUBSCRIBERFUNDOP	returns the details of a fund operation for a subscriber
VCSUBSCRVEHICUL	returns the list of major vehicles being used for fund subscription
VCSTAGE	List of investment stages
SFAACCOUNT	returns the list investing companies
SFAACCTINFOS	returns the list of company key information
SFAADDRESS	
SFACONTACT	returns the list of investing contacts

## 4.5. Step statements

**Step statements** only occur **within** programming steps. They either perform an operation or communicate information to the system. Here is the list of available statements:

Statement	Brief description	Valid in step
<a href="#">ABORT statement</a>	Stops the program	<a href="#">DATA</a>
<a href="#">ARRAY Statement</a>	Establishes a link between 1 single column and a group of columns: the single column becomes an array, and each element of the group of columns becomes an array element.	
<a href="#">BY Statement</a>	Groups variables (used with MERGE)	
<a href="#">COLUMN Statement</a>	Adds a column to a table	
<a href="#">CONTINUE Statement</a>	Exists the current control	
<a href="#">EXTEND Statement</a>	Reads data lines from input cube(s); applies only to eFront Cube programss	
<a href="#">INFILE Statement</a>	Imports an external file	
<a href="#">INLINE Statement</a>	Inserts data lines in a table	
<a href="#">MERGE Statement</a>	Merges tables	
<a href="#">OUTPUT Statement</a>	Outputs the current data line	
<a href="#">PUT Statement</a>	Writes a data value to the output	
<a href="#">RECURSE Statement</a>	Generates dependency links	
<a href="#">RETURN Statement</a>	Exists the current control	
<a href="#">SET Statement</a>	Reads data lines from input table(s)	
<a href="#">SQL Statement</a>	Includes an SQL query int the DATA step	
<a href="#">STOP Statement</a>	Stops execution of the current step	
<a href="#">TRACE Statement</a>	Writes a value to the LOG window	
<a href="#">SET Statement</a>	Reads data lines from input table(s)	<a href="#">PROC EXCEL</a>
<a href="#">TEMPLATE Statement</a>	Identifies the template to use when generating the .XLS file	
<a href="#">PICTURE Statement</a>	Creates a numeric format	<a href="#">PROC FORMAT</a>
<a href="#">VALUE Statement</a>	Creates a character format	
<a href="#">DONUT Statement</a>	Defines a donut chart	<a href="#">PROC GCHART</a>
<a href="#">DONUT3D Statement</a>	Defines a 3-dim donut chart	
<a href="#">HBAR Statement</a>	Defines a horizontal bars chart	
<a href="#">HBAR3D Statement</a>	Defines a horizontal 3-dimensional bars chart	

PIE statement	Defines a pie chart	
PIE3D statement	Defines a 3-dimensional pie chart	
TITLE statement	Defines a chart title	
VBAR statement	Defines a vertical bars' chart	
VBAR3D statement	Defines a 3-dimensional bars' chart	
PLOT statement	Defines a plots & lines chart	PROC GPLOT
BY statement	Groups variables	PROC MEANS
CLASS statement	Defines a class of values	
FIRST statement	Displays the first value of a group of values	
IRR statement	Returns the Internal Rate of Return	
JOIN statement	Groups a list of values in a single column cell	
LAST statement	Displays the last value of a group of values	
MAX statement	Returns the maximum column value	
MEAN statement	Returns the average column value	
MIN statement	Returns the minimum column value	
MULTIPLE statement	Returns the multiple of return on investment	
N statement	Returns the number of column values	
NOBS statement	Returns the number of rows in each aggregated category.	
SUM statement	Returns the sum of column values	
VAR statement	Identifies the variables on which to calculate statistics	
SET statement	Reads data lines from the input table	
TEMPLATE statement	Identifies the template to use when generating an .XLS, .DOC or XML file	PROC OFFICE
BY statement	Groups variables	PROC PRINT
EXPORT statement	Exports a table to the location specified	
FORMAT statement	Assigns a format to a column	
SUM statement	Calculates the sum of column values	
TITLE statement	Defines a table title	
VAR statement	Defines the values to print	
BY statement	Groups variables	PROC SORT
BOX statement	Defines the box above row titles	PROC TABULATE
CLASS statement	Defines a class of values	
FORMAT statement	Assigns a format to a column	
TABLE statement	Displays statistics in tabular format	
TITLE statement	Defines a table title	

<b>VAR statement</b>	Identifies the variables to calculate statistics from	<b>PROC TRANSPOSE</b>
<b>VAR statement</b>	Identifies the variables to transpose	
<b>ID statement</b>	Identifies names of transposed variables	
<b>IDLABEL statement</b>	Identifies labels of transposed variables	

## 4.5.1. ABORT statement

### 4.5.1.1. Description

Valid in: DATA steps. Stops execution of the program.

### 4.5.1.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
</ subordinate statements>;
```

```
ABORT;
```

```
RUN;
```

### 4.5.1.3. Error messages

Message	Explanation
ERREUR(0113): 'RUN' expected	The RUN statement is missing. This error is most likely to occur when you use the ABORT statement in a step other than a DATA step.

## 4.5.2. ARRAY statement

### 4.5.2.1. Description

Valid in: DATA steps. **Establishes a link** between 1 single column and a group of columns: the single column becomes an **array**, and each element of the group of columns becomes an **array element**. Therefore each array element can be accessed through an **array index** used together with the name of the array column.

### 4.5.2.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
COLUMNcol_name_1;
```

```
COLUMN col_name_n;
ARRAY col_name{} col_name_1 - col_name_n;
</ subordinate statements>;
RUN;
```

#### 4.5.2.3. Comment

- `col_name{}`: refers to the **single column** that functions as **array**.
- `col_name_1 - col_name_n`: refer to the columns that function as **array elements**.
- You must create the columns `col_name_1 - col_name_n` before applying the `ARRAY` statement. To create the column, use the `COLUMN` statement. The `ARRAY` statement doesn't create columns!

#### 4.5.2.4. Error messages

Message	Explanation

### 4.5.3. BOX statement

#### 4.5.3.1. Description

Valid in: PROC TABULATE steps. Specifies text and style elements for the empty box above row titles.

#### 4.5.3.2. Syntax

```
PROC TABULATE DATA =table_to_process (</ table_options>);
```

```
CLASS col_1 col_n;
```

```
VAR var_1 var_n;
```

```
BOX "label" | (LABEL="label" STYLE=(</ style_attributes>));
```

```
TABLE <...>;
```

```
RUN;
```

### 4.5.3.3. Comment

- CLASS, VAR, and TABLE statements are required.

### 4.5.3.4. Error messages

Message	Explanation
ERRORR(0121): {string} expected	The value assigned to BOX is wrong. Note that there is no equal sign between BOX and the character string.

## 4.5.4. BY statement

### 4.5.4.1. Description

Valid in steps: PROC MEANS, PROC PRINT, PROC SORT, and in DATA steps together with SET or MERGE statements. Identifies the variable that the programming step uses to form BY groups. You can specify more than one variable.

When using the BY statement together with the PROC PRINT step, you can choose how many groups will be opened (Expand =...) or closed (Collapse =).

### 4.5.4.2. Syntax

```
step_nameDATA = table_to_process (</ table_options>);

BY col_1(</ options>) col_n (</ options>);

</ subordinate statements>

RUN;
```

### 4.5.4.3. Comment

- According to the programming step the BY statement is used with, applicable options vary.

### 4.5.4.4. Options

Option	Valid in step	Description
--------	---------------	-------------

CLOSING	PROC PRINT	Valid if statistics are calculated for BY columns: displays only the results per group.
COLLAPSE=n	PROC PRINT	<p>n : expression that evaluates to an integer. This number represents both the number of groups to be collapsed and the direction: positive numbers count from the top, negative from the bottom.</p> <p>Should be used together with FreezeHeader and FreezeLeft options.</p> <p>Cannot be used together with the Expand option.</p> <p>If nothing is defined the usual 'all groups are expanded' applies.</p>
DESCENDING	PROC SORT	<p>Sort that decreases in value. By default sorting is ascending. A missing value is interpreted as the smallest one. DESCENDING must precede the name of the column to be sorted.</p> <p>e.g.</p> <pre>PROC SORT DATA=MYTABLE; BY DESCENDING DATE; RUN;</pre>
Expand=n	COLLAPSE=n	<p>n : expression that evaluates to an integer. This number represents both the number of groups to be expanded and the direction: positive numbers count from the top, negative from the bottom.</p> <p>Should be used together with FreezeHeader and FreezeLeft options.</p> <p>Cannot be used together with the Collapse option.</p> <p>If nothing is defined the usual 'all groups are expanded' applies.</p>
FIRSTOBS=n	PROC SORT	Sorting starts from the line being specified
GROUPFORMAT	PROC SORT	Sorting operates only on formatted values.
LASTOBS=n	PROC SORT	Sorting stops at the line being specified
NOACCENT	PROC SORT	Transform accented characters into non-accented characters. Combined with specified sorting.
UPCASE	PROC SORT	Sorting is case-insensitive. UPCASE changes temporarily all lowercase characters of the strings into uppercase characters. The sort operates only after the change.

<b>STYLE=</b>	DATA, PROC PRINT, PROC MEANS	Defines the style of the BY column
<b>LABEL=</b>	DATA, PROC PRINT, PROC MEANS	Defines the title of the BY column

- You can combine options.

#### 4.5.4.5. Error messages

Message	Explanation
column is not defined in table	The column being referenced is not defined in the data table
Unknown BY column	The variable name that follows BY is invalid

#### 4.5.5. CLASS statement

##### 4.5.5.1. Description

Valid in steps: PROC MEANS, PROC TABULATE. Defines the class of values for which to perform statistical analysis.

##### 4.5.5.2. Syntax

```
PROC PROC MEANS DATA =table_to_process (</ table_options>);
```

```
CLASS col_1 col_n;
```

```
</ subordinate statements>;
```

```
RUN;
```

##### 4.5.5.3. Comment

- CLASS applies to either numeric values or character strings. CLASS selects a group of values.
- By default, missing values in the selected columns are excluded from the analysis.
- col\_1 - col\_n: refer either to the values of a column or to the result of column values being processed, as in: CLASS FUND\_ID YEAR(REFERENCE\_DATE);

#### 4.5.5.4. Error messages

Message	Explanation
Unknown CLASS column	One of the CLASS statement parameters is wrong.
ERROR(0113): {Column name} expected	The name of the column is missing or wrong.

#### 4.5.6. COLUMN statement

##### 4.5.6.1. Description

Valid in: DATA steps. Adds a new column to a table.

##### 4.5.6.2. Syntax

DATAoutput\_table (</ table\_options>);

COLUMNcol\_name (</ options>);

</ subordinate statements>;

RUN;

##### 4.5.6.3. Comment

- To create a table from scratch, use COLUMN statements.

##### 4.5.6.4. Options

Attribute	Description
ALIGN=	Defines the alignment of the cell content. Possible values : CENTER   LEFT   RIGHT
DEFAULT=	Defines the cell content by default
FORMAT=	Defines the format to apply to the column
LABEL=	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT and PROC TABULATE steps.
LENGTH=	Possible value: integer
LOOKUP=	
TYPE=	Defines the type of the column
WIDTH=	Defines the width of the column. Possible value: integer

### 4.5.6.5. Error messages

Message	Explanation
<a href="#">ERROR: Missing or duplicate column name</a>	The name of the column is not specified, or has already been used.
Un paramètre de la macro COLUMN est invalide ou non trouvé	A parameter of the COLUMN statement is false or missing.
<a href="#">Invalid name</a>	The column option is not valid

### 4.5.7. CONTINUE statement

#### 4.5.7.1. Description

Valid in: DATA steps. Exits the current control, and jumps to the following statement.

#### 4.5.7.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
</ control_start_statement>;
```

```
CONTINUE;
```

```
</ control_end_statement>;
```

```
RUN;
```

#### 4.5.7.3. Comment

- The CONTINUE statement is applicable to [DO WHILE...LOOP WHILEcontrols](#), [FOR...TO...NEXTcontrols](#), and [FOR...IN...NEXT loops](#).

#### 4.5.7.4. Error messages

Message	Explanation
---------	-------------

## 4.5.8. DONUT statement

### 4.5.8.1. Description

Valid in: PROC GCHART steps. Defines a donut chart. The DONUT statement specifies the column(s) whose values to include in the chart.

### 4.5.8.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
```

```
DONUT col_1 (</ options>) col_n (</ options>);
```

```
RUN;
```

### 4.5.8.3. Options

Option	Description
TYPE=type_of_calculation	Defines the type of statistics' calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

### 4.5.8.4. Comment

- At least one column is required with the DONUT3D statement.

### 4.5.8.5. Output

Chart with horizontal bars.

### 4.5.8.6. Error messages

Message	Explanation

## 4.5.9. DONUT3D statement

### 4.5.9.1. Description

Valid in: PROC GCHART steps. Defines a donut chart. The DONUT3D statement specifies the column(s) whose values to include in the chart.

### 4.5.9.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
DONUT3Dcol_1 (</ options>) col_n (</ options>);
RUN;
```

### 4.5.9.3. Options

Option	Description
TYPE=type_of_calculation	Defines the type of statistics' calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

### 4.5.9.4. Comment

- At least one column is required with the DONUT3D statement.

### 4.5.9.5. Output

Chart with horizontal bars.

### 4.5.9.6. Error messages

Message	Explanation

## 4.5.10. EXTEND statement

### 4.5.10.1. Description

Valid in DATA step. Reads all the lines from 1 or more existing cubes into an output table. If you specify more than one cube, cubes are appended to each other. The EXTEND statement is similar to the [SET statement](#) with the limitation that it is impossible to add or remove rows. In comparison with the SET statement, the EXTEND statement has a much better performance as the impacted columns are not recalculated.

 Applies only to eFront Cube programs.

#### 4.5.10.2. Syntax

DATAoutput\_table (</ table\_options>);

EXTENDinput\_cube\_1 (</ table\_options>) input\_cube\_n (</ table\_options>);

RUN;

#### 4.5.10.3. Comment

- Order of concatenation : 1 to n.
- Be aware that EXTEND operates in loop-mode, therefore statements that follow the EXTEND statement are applied to each line of the output\_table.

#### 4.5.10.4. Output

eFront Report data table being enriched by one or more eFront Cube cubes.

#### 4.5.10.5. Error messages

Message	Explanation

### 4.5.11. EXPORT statement

#### 4.5.11.1. Description

Valid in: PROC PRINT steps. Exports the table to the location being specified.

#### 4.5.11.2. Syntax

PROC PRINT DATA =table\_to\_process (</ table\_options>);

EXPORTexport\_file\_1export\_file\_2;

</ subordinate statements>;

RUN;

### 4.5.11.3. Comment

- `export_file` : Example: "C:\Testreports\DealFlow.MDB";

### 4.5.11.4. Output

File

### 4.5.11.5. Error messages

Message	Explanation

## 4.5.12. FIRST statement

### 4.5.12.1. Description

Valid in: PROC MEANS steps. If several values correspond to a class of values, the FIRST statement creates a new column and displays the first corresponding value.

### 4.5.12.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) OUT = output_table;
```

```
CLASS col_name;
```

```
FIRST col_1 (</ options>) col_n (</ options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

### 4.5.12.3. Options

Option	Description
<code>LABEL="label"</code>	Defines the title of the column. By default, eFront Script displays the column name + "(last)" as column title.

#### 4.5.12.4. Comment

- CLASScol\_name:CLASS selects the group of values to which the statistical analysis applies.
- col\_1 col\_n: Defines the column(s) whose first values are to be taken into account.

#### 4.5.12.5. Error messages

Message	Explanation

### 4.5.13. FORMAT statement

#### 4.5.13.1. Description

Valid in steps: PROC PRINT, PROC TABULATE. Assigns a format to a column. A single FORMAT statement can assign the same format to several columns or different formats to different columns. The FORMAT statement can use [standard eFront Script formats](#) or formats defined using the [PROC FORMAT step](#).

#### 4.5.13.2. Syntax

step\_name;

FORMATcol\_1 format-1.col\_n format\_n.;

RUN;

#### 4.5.13.3. Comment

- A period must follow the format name.

#### 4.5.13.4. Error messages

Message	Explanation
<a href="#">Column defined more than once in a format clause</a>	The format of a column is being defined twice.
ERROR(0130)':' expected	The ':' is missing after the format name
<a href="#">Format not found</a>	The format couldn't be found

## 4.5.14. HBAR statement

### 4.5.14.1. Description

Valid in: PROC GCHART steps. Defines a chart with horizontal bars. The HBAR statement specifies the column(s) whose values to include in the chart.

### 4.5.14.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
```

```
HBARcol_1 (</ options>) col_n (</ options>);
```

```
RUN;
```

### 4.5.14.3. Options

Option	Description
TYPE=type_of_calculation	Defines the type of statistics' calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

### 4.5.14.4. Comment

- At least one column is required with the HBAR statement.

### 4.5.14.5. Output

Chart with horizontal bars.

### 4.5.14.6. Error messages

Message	Explanation

## 4.5.15. HBAR3D statement

### 4.5.15.1. Description

Valid in: PROC GCHART steps. Defines a chart with horizontal three-dimensional bars. The HBAR3D statement specifies the column(s) whose values to include in the chart.

### 4.5.15.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
HBAR3Dcol_1 (</ options>) col_n (</ options>);
RUN;
```

### 4.5.15.3. Options

Option	Description
TYPE=type_of_calculation	Defines the type of statistics' calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

### 4.5.15.4. Comment

- At least one column is required with the HBAR3D statement.

### 4.5.15.5. Output

Chart with three-dimensional horizontal bars.

### 4.5.15.6. Error messages

Message	Explanation

## 4.5.16. ID statement

### 4.5.16.1. Description

Valid in: PROC TRANSPOSE steps. Identifies the input table column whose formatted values name the variables being transposed.

### 4.5.16.2. Syntax

```
PROC TRANSPOSE DATA = table_to_process (</ table_options> </ options>);
</ subordinate statements>;
IDcol;
```

RUN;

#### 4.5.16.3. Comment

- **eFront Script** uses the values of the col being specified, as names for the new columns. For details about naming transposed variables, refer to [Naming transposed variables](#).

#### 4.5.16.4. Error messages

Message	Explanation
---------	-------------

### 4.5.17. IDLABEL statement

#### 4.5.17.1. Description

Valid in: PROC TRANSPOSE steps. Identifies the column labels for the variables being transposed.

#### 4.5.17.2. Syntax

```
PROC TRANSPOSE DATA = table_to_process (</ table_options> </ options>);
```

```
</ subordinate statements>;
```

```
IDLABELcol;
```

RUN;

#### 4.5.17.3. Comment

- **eFront Script** uses the values of the col being specified, as labels for the new columns. Labels display when using the [PROC PRINT](#) step with the LABEL option.

#### 4.5.17.4. Error messages

Message	Explanation
---------	-------------

## 4.5.18. PROC PRINT - SUM - STYLE

### 4.5.18.1. Goal

Print a table listing information about customers and items, and print the total of unit prices.

### 4.5.18.2. Features being illustrated

- PROC PRINT step
- SUM statement
- STYLE option

### 4.5.18.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a personalized format

```
PROC FORMAT;
```

```
PICTURE P_Money
```

```
LOW-HIGH = "$ 00.00";
```

```
RUN;
```

//Display data on screen

```
PROC PRINT DATA=TEST.CONTRACTS LABEL;
```

```
FORMAT UNIT_PRICE P_MONEY.;
```

```
SUM UNIT_PRICE(STYLE=(BACKGROUND=PINK));
```

```
RUN;
```

## 4.5.18.4. Result

	Contract Id	User Id	Unit Price	Quantity	
1	593	101	\$4220,10	2	
2	594	101	\$1299,00	3	
3	595	101	\$7501,00	1	
4	596	102	\$11,40	100	
5	597	102	\$6430,20	1	
6	596	103	\$7501,00	3	
7	597	101	\$6430,20	5	
8	598	105	\$20030,20	5	
			\$53423,10		

## 4.5.19. INFILE statement

### 4.5.19.1. Description

Valid in: DATA steps. Imports an external file into an eFront Report table.

Allowed file extensions are : xls, xlsx, xslm, csv and txt. It is also possible to import only an area of a .CSV file.

File path must be from on of the allowed directories defined in the package option **INFILEALLOWEDDIRECTORIES** under **pckAjxReport** package.

### 4.5.19.2. Syntax

DATAoutput\_table (</ table\_options>);

INFILE "file\_name" </ options>;

</ subordinate statements>;

RUN;

### 4.5.19.3. Options

Option	Description

AREAS= area_definition	Applies only if importing EXCEL spreadsheets. Limits the import to the area(s) being defined. The area content is fed into the columns of the data table that receives the import.  Example:  AREAS = "C7:C999;J2;A7:A999;J7:J999"; (feeds contents into 4 different data table columns)  Warning: An area of one cell means that the data will be copied for each row!
FIRSTOBS= n	Import starts from line n
INLINE	Data to be inserted are data lines. If you use the INLINE option, the <a href="#">INLINE</a> statement must precede the data lines. You don't use this option if you import a file.
NBOBS= n	Number of lines to import
SHEET=""	Identifies a sheet of an EXCEL workbook.  The expected value is either a zero-based index in the workbook (ex.: 0), or the name of the sheet to be selected ("Sheet1").
TAB   SEMICOLUMN   COMMA   SPACE	Identifies the data separator used to separate data in the data lines

#### 4.5.19.4. Comment

- file\_name is either a predefined file name or a quoted string containing the file name or character expression in parentheses referring to the file path.
- Instead of importing an external file referred to by file\_name, you can directly write data lines using the INLINE option.
- Use the INLINE option, if data lines, whose items are separated by a comma, a tab, a semicolon or a space already exist.
- Use a blank space to separate data values.

#### 4.5.19.5. Error messages

Message	Explanation
<a href="#">ERROR(0121): {filename} expected</a>	The file name being used in the INFILE statement is invalid.

## 4.5.20. INLINE statement

### 4.5.20.1. Description

Valid in: DATA steps. Inserts data lines in an existing table. Must be used together with the INFILE statement.

### 4.5.20.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
INFILE INLINE;
```

```
COLUMNcol_1 (</ options>);
```

```
COLUMNcol_2 (</ options>);
```

```
INLINE;
```

```
data_for_col_1_1data_for_col_2_1
```

```
data_for_col_1_2data_for_col_2_2
```

```
data_for_col_1_3data_for_col_2_3
```

```
data_for_col_1_ndata_for_col_2_n
```

```
;
```

```
RUN;
```

### 4.5.20.3. Comment

- Use **COLUMN** statements to create the table structure that receives the data lines.
- Use a separator (comma, semicolon, tab, space) to separate data values.

### 4.5.20.4. Error messages

Message	Explanation
---------	-------------

ERROR: Column name defined twice	The column name has been defined twice when feeding values into the data table.
Invalid value in inline data	The column type and the value are incompatible

## 4.5.21. IRR statement

### 4.5.21.1. Description

Valid in: PROC MEANS steps. Calculates the **Internal Rate of Return**.

### 4.5.21.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) OUT = output_table;
IRRcol_1 (</ options>) col_n (</ options>);
</ subordinate statements>;
RUN;
```

### 4.5.21.3. Options

Option	Description
NAME= "col_name"	Specifies the column name of the column that receives the calculation result.
LABEL="label"	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps.  Enter a bilingual label: "english_label{F}french_label".
DATE=col	Defines DATE values to be used to calculate the IRR optional; If you omit the DATE option, cashflows are considered annualized (like in the IRR EXCEL function)
RESIDUALVALUE=col	Defines the valuation as of a given date
RESIDUALDATE=col	Defines the date of the valuation

### 4.5.21.4. Comment

- col\_1- col\_n : column values must correspond to cashflow values.
- By default, if no NAME is specified, **eFront Script** generates automatically a value for NAME (col\_IRR).

#### 4.5.21.5. Error messages

Message	Explanation
---------	-------------

### 4.5.22. JOIN statement

#### 4.5.22.1. Description

Valid in: PROC MEANS steps. Groups a list of values in a single column cell.

#### 4.5.22.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) OUT = output_table;  
CLASS col_1;  
JOIN col_2 (NAME=column_name DELIMITER="<delimiter>");  
</ subordinate statements>;  
RUN;
```

#### 4.5.22.3. Comment

- col\_1: Selects the column for which values are to be regrouped.
- col\_2: Selects the column that receives regrouped values.
- DELIMITER= "<delimiter>": Specifies the delimiter to put between values, such as ",".
- NAME=column\_name: Specifies the name of the column that receives joint values.  
The name by default=col\_2\_JOIN.

#### 4.5.22.4. Error messages

Message	Explanation
---------	-------------

## 4.5.23. LAST statement

### 4.5.23.1. Description

Valid in: PROC MEANS steps. If several values correspond to a class of values, the LAST statement creates a new column and displays the last corresponding value.

### 4.5.23.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) OUT = output_table;
CLASS col_name;
LAST col_1 (</ options>) col_n (</ options>);
</ subordinate statements>;
RUN;
```

### 4.5.23.3. Options

Option	Description
NAME= col_name	Defines the name of the column to create
LABEL="label"	Defines the title of the column to be displayed. By default, eFront Script displays the column name of the column whose last value should be taken + "(last)" as column title. If a NAME= has been defined, eFront Script displays only the name being defined without adding "(last)".

### 4.5.23.4. Comment

- CLASS col\_name: CLASS selects the group of values to which the statistical analysis applies.
- col\_1 col\_n: Defines the column(s) whose last values are to be taken into account.
- Ex. :

...  
LAST TRANSACTION\_STAKE(NAME=TRANSACTION\_STAKE LABEL="% stake  
(trans){F}%détention (trans));

...

## 4.5.24. MAX statement

### 4.5.24.1. Description

Valid in: PROC MEANS steps. Returns the maximum value of a column.

### 4.5.24.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) OUT = output_table;  
MAXcol_1 (</ options>) col_n (</ options>);  
</ subordinate statements>;  
RUN;
```

### 4.5.24.3. Options

Option	Description
NAME="col_name"	Specifies the column name of the column that receives the calculation result.
LABEL="label"	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps.

### 4.5.24.4. Comment

- By default, if no NAME is specified, **eFront Script** generates automatically a value for NAME (col\_MAX).

### 4.5.24.5. Error messages

Message	Explanation
Unknown MAX column	The column referred to by the MAX statement is invalid.

## 4.5.25. MEAN statement

### 4.5.25.1. Description

Valid in: PROC MEANS steps. Returns the average of the values of a column.

### 4.5.25.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) OUT = output_table;
MEAN col_1 (</ options>) col_n (</ options>);
</ subordinate statements>;
RUN;
```

### 4.5.25.3. Options

Option	Description
NAME= "col_name"	Specifies the column name of the column that receives the calculation result.
LABEL="label"	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps.

### 4.5.25.4. Comment

- By default, if no NAME is specified, **eFront Script** generates automatically a value for NAME (col\_MEAN).

### 4.5.25.5. Error messages

Message	Explanation
Unknown MEAN column	The column referred to by the MEAN statement is invalid.

## 4.5.26. MERGE statement

### 4.5.26.1. Description

Valid in: DATA steps. Merges lines being read simultaneously from at least 2 input tables, into one single line of the output table. Merge criteria : a common variable.

### 4.5.26.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
MERGEinput_table_1 (</ table_options>) input_table_n (</ table_options>);
```

```
BYcol_1 (</ options>) col_n (</ options>);
```

```
</ subordinate statements>;
```

```
RUN;
```

#### 4.5.26.3. Comment

- MERGE requires at least two input tables.
- The BY-columns must exist in all the tables to merge.
- You can apply WHERE conditions to BY columns.
- When applying a WHERE condition to a BY column, the order in which input tables appear in the MERGE statement is important. For details, refer to the example : [Merge two tables with a condition applied to the common variable](#).
- To select lines to include in the merge result, use [IF...THEN...ELSE](#) controls. To express conditions, you might need to know if the data line is part of the tables to merge. Use the table option [IN](#) to flag the data as being part of the table, and build conditions based upon the flag state:

```
MERGEinput_table_1 (IN=flag_1) input_table_2 (IN=flag_2);
```

```
BYcol;
```

```
Ifflag_1 <operator> flag_2 THEN
```

```
</ subordinate statements>;
```

```
OUTPUT;
```

```
END;
```

#### 4.5.26.4. Example of the use of the IN flag together with the MERGE step

The IN= allows one to flag the table from which a row comes from in a merge operation. If used it must be the FIRST element on qualifying a table and then may be tested.

Consider a merge below:

```
DATAnewtable;
```



```
MERGE tableone tabletwo;
```

```
Bycolumn;
```

```
RUN;
```

Assume you only wish to keep rows if they come from BOTH tables, you could write:

```
DATA newtable;
```

```
MERGE tableone(IN=IN1) tabletwo(IN=IN2);
```

```
Bycolumn;
```

```
If (IN1 and IN2) then Output;
```

```
End;
```

```
RUN;
```

#### 4.5.26.5. Error messages

Message	Explanation
ERROR(0113): Table name expected	The table name has been left out in the MERGE statement, or erroneously assigned using an "=" sign.
Cannot register table	The table that should be read, cannot be found.

#### 4.5.27. MIN statement

##### 4.5.27.1. Description

Valid in: PROC MEANS steps. Returns the minimum value of the values of a column.

##### 4.5.27.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) OUT = output_table;
```

```
MINcol_1 (</ options>) col_n (</ options>);
```

```
</ subordinate statements>;
```

RUN;

### 4.5.27.3. Options

Option	Description
NAME= "col_name"	Specifies the column name of the column that receives the calculation result.
LABEL="label"	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps.

### 4.5.27.4. Comment

- By default, if no NAME is specified, **eFront Script** generates automatically a value for NAME (col\_MIN).

### 4.5.27.5. Error messages

Message	Explanation
Unknown MIN column	The column referred to by the MIN statement is invalid.

## 4.5.28. MULTIPLE statement

### 4.5.28.1. Description

Valid in: PROC MEANS steps. Returns the multiple of the return on investment.

### 4.5.28.2. Syntax

PROC MEANS DATA = table\_to\_process (</ table\_options>) OUT = output\_table;

MULTIPLEcol\_1 (</ options>) col\_n (</ options>);

</ subordinate statements>;

RUN;

### 4.5.28.3. Options

Option	Description
--------	-------------

NAME= "col_name"	Specifies the column name of the column, that receives the calculation result.
DATE=col	Defines the cell content by default
AMOUNT=col	Defines the format to apply to the column
LABEL="label"	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps.

#### 4.5.28.4. Comment

- By default, if no NAME is specified, **eFront Script** generates automatically a value for NAME (col\_MULTIPLE).
- DATE and AMOUNT options are required.

#### 4.5.28.5. Error messages

Message	Explanation
Unknown MULTIPLE column	The column referred to by the MULTIPLE statement is invalid.

### 4.5.29. N statement

#### 4.5.29.1. Description

Valid in: PROC MEANS steps. Counts the number of variable instances (non missing values).

#### 4.5.29.2. Syntax

PROC MEANS DATA = table\_to\_process (</ table\_options>) OUT = output\_table;

Ncol\_1 (</ options>) col\_n (</ options>);

</ subordinate statements>;

RUN;

#### 4.5.29.3. Options

Option	Description
NAME="col_name"	Specifies the column name of the column that receives the calculation result.

<code>LABEL="label"</code>	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps.
----------------------------	--

#### 4.5.29.4. Comment

- By default, if no NAME is specified, **eFront Script** generates automatically a value for NAME (col\_N).

#### 4.5.29.5. Error messages

Message	Explanation
<code>Unknown N column</code>	The column referred to by the N statement is invalid.

### 4.5.30. NOBS statement

#### 4.5.30.1. Description

Valid in the PROC MEANS step. Counts the number of rows in each aggregated category.

#### 4.5.30.2. Syntax

```
PROC MEANS DATA = table_to_process (</ table_options>) </ options>;
```

```
CLASScol_1;
```

```
NOBScol_name;
```

```
</ subordinate statements>;
```

```
RUN;
```

#### 4.5.30.3. Comment

- CLASScol\_1:CLASS selects the group of values to which the category aggregation applies.
- col\_name: the name of the table that contains the aggregated values.

#### 4.5.30.4. Output

A column containing the number of rows in the aggregated category.

#### 4.5.30.5. Error messages

Message	Explanation

### 4.5.31. OUTPUT statement

#### 4.5.31.1. Description

Valid in: DATA steps. Writes the current data line from within the DATA step to the output table.

#### 4.5.31.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
</ subordinate statements>;
```

```
OUTPUT;
```

```
RUN;
```

#### 4.5.31.3. Comment

- The OUTPUT statement is necessary within a DATA step if you use no SET, MERGE or INFILE statement. If no **SET**, **MERGE** or **INFILE** statement is used, **eFront Script** writes data to the **output\_table** only at the end of the DATA step. Therefore, if a loop is being used, only the last data line processed by the loop is written to the **output\_table**, and only if the **\_OUTPUT\_** flag = TRUE. If you use the OUTPUT statement, all the lines are written to the **output\_table**.
- As soon as a DATA step reads an OUTPUT statement, the **\_OUTPUT\_** flag is set to FALSE.
- If you want to write each line to the output table, use the OUTPUT statement. If you want to write lines only under specific conditions, use the **\_OUTPUT\_** flag.
- To write several times the same line to the output table, multiply OUTPUT statements.

#### 4.5.31.4. Error messages

Message	Explanation
---------	-------------

### 4.5.32. PICTURE statement

#### 4.5.32.1. Description

Valid in: PROC FORMAT steps. Specifies a template (= picture) for formatting numeric values.

#### 4.5.32.2. Syntax

PROC FORMAT;

PICTURE`picture_name`

`value_or_range_1 = "picture"`

`value_or_range_n = "picture"`

;

RUN;

#### 4.5.32.3. Comment

- The PICTURE statement creates a format for numeric values. You can use several PICTURE statements in a PROC FORMAT step.

#### 4.5.32.4. Error messages

Message	Explanation
ERROR: Format defined twice	The same picture has been defined twice
ERROR(0113): {Format name} expected	The format name is missing in the PICTURE statement
ERROR: Invalid definition	The picture definition is invalid

## 4.5.33. PIE statement

### 4.5.33.1. Description

Valid in: PROC GCHART steps. Defines a pie chart. The PIE statement specifies the columns(s) whose values to include in the pie chart. Each pie slice represents the value of the chart statistic for a particular column in relation to the total chart statistic for all columns.

### 4.5.33.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
```

```
PIEcol_1 (</ options>) col_n (</ options>);
```

```
RUN;
```

### 4.5.33.3. Options

Option	Description
TYPE=type_of_calculation	Defines the type of statistics' calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

### 4.5.33.4. Comment

- At least one column is required with the PIE statement.

### 4.5.33.5. Output

Pie chart

### 4.5.33.6. Error messages

Message	Explanation

## 4.5.34. PIE3D statement

### 4.5.34.1. Description

Valid in: PROC GCHART steps. Defines a 3-dimensional pie chart. The PIE3D statement specifies the columns(s) whose values to include in the pie chart. Each pie

slice represents the value of the chart statistic for a particular column in relation to the total chart statistic for all columns.

#### 4.5.34.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
PIE3Dcol_1 (</ options>) col_n (</ options>);
RUN;
```

#### 4.5.34.3. Options

Option	Description
TYPE=type_of_calculation	The type of statistics calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

#### 4.5.34.4. Comment

- At least one column is required with the PIE3D statement.

#### 4.5.34.5. Output

Pie chart

#### 4.5.34.6. Error messages

Message	Explanation

### 4.5.35. PLOT statement

#### 4.5.35.1. Description

Valid in: PROC GPLOT steps. Defines a plots & lines chart. The PLOT statement specifies the column(s) whose values to include in the chart.

#### 4.5.35.2. Syntax

```
PROC GPLOT DATA =table_to_process (</ table_options>);
```

```
PLOT col_1*col2 (</ options>) col_n*col_m (</ options>);
```

RUN;

### 4.5.35.3. Options

Option	Description
INTERPOL=line	Defines the type of line. Possible values are: SPLINE   JOIN   STEP
SYMBOL=plot	Defines the type of plot. Possible values are: SQUARE   DIAMOND   PLUS   X   CIRCLE   STAR   TRIANGLE   RIGHT TRIANGLE   LEFT TRIANGLE   INVERTED TRIANGLE   POLYGON   POLYGON2

### 4.5.35.4. Comment

- At least one couple col\_1\*col\_2 is required.

### 4.5.35.5. Output

Plots & lines chart.

### 4.5.35.6. Error messages

Message	Explanation

## 4.5.36. PUT statement

### 4.5.36.1. Description

Valid in: DATA and PROC PRINT steps. Writes data values to a special buffer from which they can be written to the data component.

### 4.5.36.2. Syntax

```
DATAoutput_table (</ table_options>);
```

</ subordinate statements>;

```
PUT'argument_1"argument_n';
```

RUN;

### 4.5.36.3. Comment

- argument : variable, string, number.
- You can style the PUT statement output, and use **HTML** tags as an argument. For example: PUT "table<TR><TD>";
- The PUT statement without arguments, writes the current output line to the current location, even if the current output line is blank.
- To insert an empty line, use PUT CRLF();

### 4.5.36.4. Output

Direct output

### 4.5.36.5. Error messages

Message	Explanation
---------	-------------

## 4.5.37. RECURSE statement

### 4.5.37.1. Description

Valid in: DATA steps. Generates dependency links from a table that includes Parent/Child informations.

### 4.5.37.2. Syntax

DATAoutput\_table (</ table\_options>);

RECURSEinput\_table\_1 (</ options>) input\_table\_2 (</ options>);

COLUMNcol\_name (</ options>);

</ subordinate statements>;

RUN;

### 4.5.37.3. Comment

- The output\_table must include parent/child information.

- The **COLUMN** statement is required to list dependency links generated by the RECURSE statement.
- Use the **\_LEVEL\_** variable to display levels of hierarchy.
- You cannot apply table options to a RECURSEinput\_table.

#### 4.5.37.4. Options

Option	Description
VAR=child_col	Specifies the child_col when establishing a link between a child and a parent
PARENT=parent_col	Specifies the parent_col when establishing a link between a child and a parent
START =<expression>	Determines the point from which to calculate parent-child links. If you don't include the START option, child-parent link calculation starts from the first parent value

- VAR and PARENT options are required

#### 4.5.37.5. Error messages

Message	Explanation

### 4.5.38. RETURN statement

#### 4.5.38.1. Description

Valid in: DATA steps. Exits the current control, and jumps to the following statement.

#### 4.5.38.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
</ control_start_statement>;
```

```
RETURN;
```

```
</ control_end_statement>;
```

```
RUN;
```

### 4.5.38.3. Comment

- The RETURN statement is applicable to DO WHILEcontrols, FOR...TO...NEXTcontrols, IF...THEN...ELSEcontrols, SELECT...WHEN...THENcontrols.

### 4.5.38.4. Error messages

Message	Explanation
---------	-------------

## 4.5.39. SET statement

### 4.5.39.1. Description

Valid in steps: DATA, PROC OFFICE, PROC EXCEL. Reads all the lines from 1 or more existing tables into an output table. If you specify more than one table, tables are appended to each other, with the same columns being regrouped.

### 4.5.39.2. Syntax

DATAoutput\_table (</ table\_options>);

SETinput\_table\_1 (</ table\_options>) input\_table\_n (</ table\_options>);

RUN;

### 4.5.39.3. Comment

- If you don't specify any input\_table, the SET statement reads the lines from the most recently created table.
- Order of concatenation : 1 to n.
- To copy a table, use the SET statement.
- To copy a view into a DATA step, use the SET statement.
- Be aware that SET operates in loop-mode, therefore statements that follow the SET statement are applied to each line of the output\_table.
- To select lines to include in the SET result, use IF...THEN...ELSE controls. To express conditions, you might need to know if the data line is part of the input tables. Use the table option IN to flag the data as being part of the table, and build conditions based upon the flag state:

```
SETinput_table_1 (IN=flag_1) input_table_2 (IN=flag_2);
```

```
Ifflag_1 <operator> flag_2 THEN
```

```
</ subordinate statements>;
```

```
OUTPUT;
```

```
END;
```

#### 4.5.39.4. Error messages

Message	Explanation
ERROR(0113): Table name expected	The table name has been left out in the SET statement, or erroneously assigned using an "=" sign.
Cannot register table :	The table that should be read, cannot be found.

#### 4.5.40. SQL statement

##### 4.5.40.1. Description

Valid in: DATA steps. Allows you to integrate an SQL query directly in a DATA step. The SQL query only operates on the current **eFront** database. Be careful: with this function you can write an SQL statement which completely overpasses the regions and access rights, therefore, in most cases, it is better to create a view first, and then copy it into a data table.

##### 4.5.40.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
SQL "<sql_query>" </ options>;
```

```
COLUMNcol_name;
```

```
COLUMNcol_name;
```

```
RUN;
```

### 4.5.40.3. Options

Option	Description
CONNECTION=connection_string	By default, the SQL query being defined extracts data from the current eFront application. To access another data base, specify the data base connection string

### 4.5.40.4. Comment

- COLUMN statements are necessary to create the table that receives the data being retrieved by the SQL statement.

### 4.5.40.5. Error messages

Message	Explanation

## 4.5.41. STOP statement

### 4.5.41.1. Description

Valid in: DATA steps. Stops execution of the current step, starts execution of the next step.

### 4.5.41.2. Syntax

DATAoutput\_table (</ table\_options>);

</ subordinate statements>;

STOP;

RUN;

### 4.5.41.3. Error messages

Message	Explanation
ERREUR(0113): 'RUN' attendu	The RUN statement is missing. This error is most likely to occur when you use the STOP statement in a step other than a DATA step.

## 4.5.42. SUM statement

### 4.5.42.1. Description

Valid in steps: PROC MEANS, PROC PRINT. Sums up the values of a column. The Sum statement creates a new column to display the result.

### 4.5.42.2. Syntax

```
step_nameDATA = table_to_process (</ table_options>);

SUMcol_1 (</ options>) col_n (</ options>);

</ subordinate statements>

RUN;
```

### 4.5.42.3. Options

Option	Description
NAME="col_name"	Specifies the column name of the column that receives the calculation result.
FORMAT=format.	Specifies the format to apply to the column.
LABEL="label"	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps.
STYLE= (</ style_attributes>)	Specifies the style to apply to the column.

### 4.5.42.4. Comment

- By default, if no NAME is specified, **eFront Script** generates automatically a value for NAME (col\_SUM).
- When using SUM in a PROC PRINT step, the only available option is STYLE.
- When using SUM within the VAR statement in a PROC PRINT step, SUM must be preceded by a period.
- When using SUM in a PROC MEANS step, all options are available, except for STYLE.

#### 4.5.42.5. Error messages

Message	Explanation
Unknown SUM column	The column referred to by the SUM statement is invalid

#### 4.5.43. TABLE statement

##### 4.5.43.1. Description

Valid in: PROC TABULATE step. Displays descriptive statistics in tabular format, using some or all of the variables of a data table.

##### 4.5.43.2. Syntax

```
PROC TABULATE DATA = table_to_process (</ table_options>);
```

```
CLASS col_1 col_n;
```

```
VAR var_1 var_n;
```

```
TABLE </ line_structure>,
```

```
</ column_structure>*
```

```
(</ functions>);
```

```
RUN;
```

##### 4.5.43.3. Line structure - Column structure

Two ways of specifying line or column structures:

1: var\_1 = "label\_1" var\_n = "label\_n"

2: var\_1 = (LABEL = "label\_1" STYLE(item\_to\_be\_styled = </ style\_attributes>)) var\_n = (LABEL = "label\_n" STYLE(item\_to\_be\_styled = </ style\_attributes>))

- If no "label" is being specified, the default label is the variable name.
- Place , after the line structure.
- Place \* after the column structure.

- **ALL** summarizes the information of the variable: var = "label\_1" ALL= "Total"
- **STYLE** applies a particular style to a table item.
- If applied to a variable other then ALL, valid values for item\_to\_be\_styled are DATA | CLASS | HEADER.
- If applied to ALL, valid values for item\_to\_be\_styled are DATA | HEADER.

#### 4.5.43.4. Functions

Two ways of specifying functions:

1: function\_1 = "label\_1" function\_n = "label\_n";

2: function\_1 = (LABEL = "function\_1" FORMAT = format.STYLE(item\_to\_be\_styled = </ style\_attributes>)) function\_n = (LABEL = "label\_n" FORMAT = format.STYLE(item\_to\_be\_styled = </ style\_attributes>));

- **STYLE** applies a particular style to the value returned by the function.
- Valid values for item\_to\_be\_styled are DATA | CLASS | HEADER.
- Valid values for function are N | MAX | MIN | MEAN | STD | SUM.
- If you don't specify VAR nor a statistical option, TABLE counts values (N).

#### 4.5.43.5. Comment

- CLASS selects a group of values.
- VAR identifies the numeric variables on which to calculate statistics.
- TABLE describes the table that receives the calculation results.
- By default, the TABLE statement produces columns.
- To indicate that variable values are written into lines, place a comma after the variables.
- To indicate that variable values are written into columns, place a star after the variables.
- All variables in the TABLE statement must appear in either the VAR statement or the CLASS statement.

#### 4.5.43.6. Error messages

Message	Explanation
---------	-------------

Cannot mix columns and statistics at the same level	It might identify an erroneous use of column/line identifiers.
Invalid TABLE statistic: No calc variable defined	The calculation variable in the TABLE statement has been left out.
Unknown TABLE column	One of the TABLE statement parameters is not recognized

## 4.5.44. TEMPLATE statement

### 4.5.44.1. Description

VALID in steps: PROC OFFICE, PROC EXCEL. Calls the template file to be used to create a WORD, and EXCEL or an XML report.

### 4.5.44.2. Syntax

step\_name file\_name (</ table\_options>);

TEMPLATE &file\_name;

</ subordinate statements>;

RUN;

### 4.5.44.3. Comment

- You must prefix the file\_name with &.
- You can use the eFront Report document editor to build XLS, and DOC templates.
- For the moment, the eFront Report document editor doesn't manage the XML templates yet. Therefore you can only generate XML reports through the launch of the eFront Script program.

### 4.5.44.4. Error messages

Message	Explanation
ERROR(0121): {template-name} expected	The template file name is missing, or invalid. Remember that the file name has to be a quoted string.
Cannot find Office template file	The template file name is invalid, or the file cannot be found at the location being specified.
Invalid Office template	The file being defined is not a valid OFFICE template

Office template file is not defined	No template file has been defined.
-------------------------------------	------------------------------------

## 4.5.45. TITLE statement

### 4.5.45.1. Description

Valid in steps: PROC PRINT, PROC TABULATE, PROC GCHART. Defines the table title to display.

### 4.5.45.2. Syntax

```
step_nameDATA =table_to_process (</ table_options>);
```

```
TITLE"title";
```

```
</ subordinate statements>;
```

```
RUN;
```

### 4.5.45.3. Comment

- 10 hierarchical levels of titles are available (TITLE1, TITLE2,...,TITLE10).

### 4.5.45.4. Error messages

Message	Explanation
ERROR(0105): {EOF} expected	The TITLE statement is not included in a valid PROC step.
ERROR(0121): {string} expected	The value assigned to the TITLE statement is invalid. Note : there is no equal sign!

## 4.5.46. TRACE statement

### 4.5.46.1. Description

Valid in DATA step, as well as **outside** steps. Allows to write a value to the **LOG** window, This value can be an **alert**, the content of a **column**, or the content of **macro variables**. Use this statement to **debug eFront Script** programs.

## 4.5.46.2. Syntax

TRACEvalue;

## 4.5.46.3. Comment

- value: "string" | "number" | variable
- You can use the TRACE statement to **evaluate program execution time**: use TRACE statements in the beginning and in the end of the program.
- You can activate Trace statements

## 4.5.46.4. Error messages

Message	Explanation

## 4.5.47. TWR statement

### 4.5.47.1. Description

Valid in: PROC MEANS steps. Calculates the **Time Weighted Return**.

### 4.5.47.2. Syntax

PROC MEANS DATA = table\_to\_process (</ table\_options>) OUT = output\_table;

TWR col\_1 (</ options>) col\_n (</ options>);

</ subordinate statements>;

RUN;

### 4.5.47.3. Options

Option	Description
NAME = "col_name"	Specifies the column name of the column that receives the calculation result.
LABEL = "label"	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT steps. Enter a bilingual label: "english_label{F}french_label".
DATE = col	Defines the date of each cashflow.

Option	Description
RESIDUALVALUE = col	Defines the valuation as of any given date.
RESIDUALDATE = col	Defines the date of the valuation.
NAV = col	Define the column containing the NAV. If NAV is specified, each NAV operation is used to define the TWR periods. The period parameter is then ignored.
GROUPBY = col	When defining a GROUPBY column, we group cashflows using this column, calculate the TWR independently for each groupby item, then return a geometric mean of all the TWR.
MONTH = num	Duration of the TWR calculation in month. For instance, MONTH=3 will calculate the 3M TWR (TWR over the last 3 months). When both MONTH, DAY and TWRTODATE are unspecified, we calculate an inception TWR.
DAY = num	Period of the TWR calculation in days. If MONTH is specified, the DAY is ignored. For instance, DAY=10 will calculate a 10-day TWR (TWR over the last 10 days).
FREQUENCY	Analysis frequency. If DAY is specified, then the frequency unit is days, otherwise it is in months. If unspecified, the default frequency is 3 months.
ANNUALIZED	When defined, the TWR is annualized.
STARTOFDAY	Modify the calculation of the Dietz Factor. See the Modified Dietz section below.
USEFIRSTPERIOD	When defined, keep the initial partial period. See the section below on "start date of the analysis".
TWRTODATE = "value"	TWRTODATE can be set to QTD or YTD, to calculate QTD TWR or YTD TWR. When using TWRTODATE, neither MONTH nor DAY should be used.
NMSUBPERIOD	If the TWR calculated over a sub-period returns a non-meaningful number (typically when the investment is fully exited), then do not return a number. This parameter is mainly useful in combination with GROUPBY.
EXITDATE = col	Exit date of the investment. For a single TWR (without GROUPBY), this column should always show the same value. When using a GROUPBY criteria, this column should show one value for each groupby item.
USEALLSERIES	Only relevant when using GROUPBY. See the start date note below.
ALLOWRECENTINVESTMENT	Only relevant when using MONTH or DAY. See the start date note below.
CUSTOMDIETZFACT OR = num	Custom Dietz Factor. If this option is set, then ALLOWRECENTINVESTMENT is also set. See the modified dietz section below for details.

#### 4.5.47.4. Comment

This function calculates a Time weighted return using the modified dietz method.

#### Modified Dietz TWR:

The modified Dietz TWR calculation split the time into subperiods. For each subperiod, we calculate a TWR return using the end market value (EMV), the beginning market value (BMV), and the list of cashflows in the period:

$$TWR = \frac{(EMV - (BMV + CF))}{(BMV + CF_w)} \quad (1)$$

$$CF_w = \sum_i CF_i \cdot DietzFactor_i \quad (2)$$

The dietz factor calculation depends on the parameter provided.

- If CUSTOMDIETZFACTOR is set, then:

$$DietzFactor_i = CUSTOMDIETZFACTOR \quad (3)$$

for all cashflows.

- If STARTOFDAY is set, then:

$$DietzFactor_i = \frac{PeriodEndDate - CFDate_i}{PeriodLength} \quad (4)$$

- Otherwise:

$$DietzFactor_i = \frac{PeriodEndDate - CFDate_i + 1}{PeriodLength} \quad (5)$$

## Start date of the analysis

The TWR start date is defined by the following parameters:

- MONTH
- DAY
- TWRTODATE
- ALLOWRECENTINVESTMENT
- USEALLSERIES
- USEFIRSTPERIOD

When using the option MONTH or DAY, we calculate the TWR of the investment over a given period of time. The start date of the analysis is then calculated as [Residual

Date] – [Period of time provided]. If we define TWRTODATE as QTD or YTD, then the start date of the analysis is calculated as respectively QuarterStart([Residual Date]) or YearStart([Residual Date]). If none of those are defined, the start date is set to the date of the first cashflow.

If the first operation of the investment is after the start date, then:

- If USEALLSERIES is set, then we keep the start date before the first operation date.
- If USEALLSERIES is not set, then:
  - If ALLOWRECENTINVESTMENT is **not** set, we return a blank.
  - If ALLOWRECENTINVESTMENT is set, then we set the start date to the date of the first operation.

Example:

If MONTH=12, and residual date = 30/06/2022, then we calculate a 12-month TWR, with a start date of 30/06/2021.

If the first investment operation is on 30/09/2021, then:

- If ALLOWRECENTINVESTMENT is not set, we return a blank.
- If ALLOWRECENTINVESTMENT is set, we return a 9-month TWR (starting of 30/09/2021).

## USEFIRSTPERIOD

We acknowledge the typo, which we'll endeavor to fix soon. By default (with USEFIRSTPERIOD not set), we remove any partial period at the start of the investment. This may happen if:

- The value for MONTH/DAY is not a multiple of FREQUENCY. As a best practice, MONTH/DAY, when set, should always be a multiple of FREQUENCY.
- No value is specified for MONTH/DAY (so we calculate an inception TWR) and the first operation date does not fall on a multiple of the FREQUENCY. As an example, if FREQUENCY=3, we will start the analysis on the first quarter after the first cashflow of the investment.

col\_1- col\_n : Column values must correspond to cashflow values.

By default, if no NAME is specified, **eFront Script** automatically generates a value for NAME (col\_IRR).

#### 4.5.47.5. Error messages

Message	Explanation
---------	-------------

### 4.5.48. VALUE statement

#### 4.5.48.1. Description

Valid in steps : PROC FORMAT. Creates a character format, i.e. specifies the character string that replaces stored variable values.

#### 4.5.48.2. Syntax

PROC FORMAT;

  VALUE \$format\_name

    value\_or\_range\_1 = "value\_displayed\_1"

    value\_or\_range-n = "value\_displayed\_n"

  ;

RUN;

#### 4.5.48.3. Comment

- Displayed values are always character strings, regardless of whether you are creating a character or numeric format.
- Displayed values must be enclosed in simple or double quotations marks.
- value specifications are case-sensitive.
- The VALUE statement creates a format. You can use several VALUE statements in a PROC FORMAT step.

#### 4.5.48.4. Error messages

Message	Explanation
ERROR(0113): {Format name} expected	The format name is missing in the VALUE statement
ERROR: Format defined twice	The same format has been defined twice
ERROR: Invalid definition	The value definition is invalid

#### 4.5.49. VAR statement

##### 4.5.49.1. Description

Valid in steps : PROC TABULATE, PROC PRINT, PROC TRANSPOSE, PROC MEANS, PROC EFRONTBLOB, PROC CONVERTCURR.

- In PROC TABULATE: identifies the numeric variables on which to calculate statistics. The values of the variables can be continuous or discrete.
- In PROC PRINT: identifies the variables to print and determines their order. Print order = order of appearance in statement.
- In PROC TRANSPOSE: identifies the variables to transpose from columns into lines.
- In PROC MEANS: identifies the numeric variables on which to calculate statistics.
- In PROC EFRONTBLOB: identifies the columns of the input table (DATA=xxx) to be kept in the eFront Report output table.
- In PROC CONVERTCURR: identifies the columns of the input cube to be converted, using the specified FX rates.

##### 4.5.49.2. Syntax valid in a PROC TABULATE step

PROC TABULATE DATA = table\_to\_process (</ table\_options>;

CLASS col\_1 col\_n;

VAR var\_1 var\_n;

</ subordinate statements>;

RUN;

##### 4.5.49.3. Syntax valid in a PROC MEANS step

PROC MEANS DATA = table\_to\_process (</ table\_options>;

```
CLASScol_1col_n;  
VARvar_1var_n;  
</ subordinate statements>;  
RUN;
```

#### 4.5.49.4. Syntax valid in a PROC TRANSPOSE step

```
PROC TRANSPOSE DATA = table_to_process (</ table_options>);
```

```
VARvar_1var_n;  
</ subordinate statements>;  
RUN;
```

#### 4.5.49.5. Syntax valid in a PROC PRINT step

```
PROC PRINT DATA = table_to_print (</ table_options>);
```

```
VARvar_1 (</ options>) var_n (</ options>)  
</ functions>;  
RUN;
```

#### 4.5.49.6. Syntax valid in a PROC EFRONTBLOB step

```
PROC EFRONTBLOB </options>;
```

```
VARvar_1var_n;  
RUN;
```

#### 4.5.49.7. Syntax valid in a PROC CONVERTCURR step

```
PROC CONVERTCURR </options>;
```

```
VARvar_1 (</ options>) var_n (</ options>)  
RUN;
```

## 4.5.49.8. Options

Option	Description
<code>FORMAT=format.</code>	Specifies the format to apply to the column.
<code>NAME=format</code>	Specified the name of the target column
<code>LABEL="label"</code>	Defines the title of the column. By default, eFront Script displays the column name as column title. To display labels as column titles, add the option LABEL to PROC PRINT and PROC TABULATE steps.
<code>STYLE= (&lt;/style_attributes&gt;)</code>	Specifies the style to apply to the column.

## 4.5.49.9. Functions

Function	Description
<code>.IRR</code>	<code>.IRR(DATE=colAMOUNT=colFORMAT=format.LABEL="label"STYLE(DATA)= &lt;/style_attributes&gt; STYLE(TOTAL) = &lt;/ style_attributes&gt;)</code>
<code>.MAX</code>	<code>.MAX(VAR=colFORMAT=format.LABEL="label"STYLE(DATA)= &lt;/ style_attributes&gt; STYLE(TOTAL) = &lt;/ style_attributes&gt;)</code>
<code>.MEAN</code>	<code>.MAX(VAR=colFORMAT=format.LABEL="label"STYLE(DATA)= &lt;/ style_attributes&gt; STYLE(TOTAL) = &lt;/ style_attributes&gt;)</code>
<code>.MIN</code>	<code>.MIN(VAR=colFORMAT=format.LABEL="label"STYLE(DATA)= &lt;/ style_attributes&gt; STYLE(TOTAL) = &lt;/ style_attributes&gt;)</code>
<code>.MULTIPLE</code>	<code>.MULTIPLE(LABEL="label"VAR=colRESIDUALVALUE=colFORMAT=format.STYLE(DATA)= &lt;/style_attributes&gt; STYLE(TOTAL) = &lt;/ style_attributes&gt;)</code>
<code>.SUM</code>	Calculates the sum of a column

## 4.5.49.10. Comment

- A period precedes all function used within the VAR statement.

## 4.5.49.11. Error messages

Message	Explanation
<code>ERROR(0113): {Column name} expected</code>	No variable has been defined in the VAR statement
Invalid TABLE statistic: No calc variable defined	The calculation variable defined in the VAR statement is wrong.
<code>ERREUR: Option de variable invalide :</code>	The option used to closer define VAR, is invalid.

## 4.5.50. VBAR statement

### 4.5.50.1. Description

Valid in: PROC GCHART steps. Defines a chart with vertical bars. The VBAR statement specifies the column(s) whose values to include in the chart.

### 4.5.50.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
```

```
VBARcol_1 (</ options>) col_n (</ options>);
```

```
RUN;
```

### 4.5.50.3. Options

Option	Description
TYPE=type_of_calculation	Defines the type of statistics' calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

### 4.5.50.4. Comment

- At least one column is required with the VBAR statement.

### 4.5.50.5. Output

Chart with vertical bars.

### 4.5.50.6. Error messages

Message	Explanation

## 4.5.51. VBAR3D statement

### 4.5.51.1. Description

Valid in: PROC GCHART steps. Defines a chart with vertical three-dimensional bars. The VBAR3D statement specifies the column(s) whose values to include in the chart.

### 4.5.51.2. Syntax

```
PROC GCHART DATA =table_to_process (</ table_options>);
```

```
VBAR3Dcol_1 (</ options>) col_n (</ options>);
```

```
RUN;
```

### 4.5.51.3. Options

Option	Description
TYPE=type_of_calculation	Defines the type of statistics' calculation to apply to the pie column. Possible values are: N   SUM   PCTN   PCTSUM   MEAN
SUMVAR=col	Calculates the sum for each column value

### 4.5.51.4. Comment

- At least one column is required with the VBAR3D statement.

### 4.5.51.5. Output

Chart with three-dimensional vertical bars.

### 4.5.51.6. Error messages

Message	Explanation

## 4.6. Options

Options can be used together with [statements](#). They allow to customize the statements. Some options stand alone, other options need a value being assigned to them.

Options can apply to the entire table or only to individual columns:

### Table Options:

- [Table options](#)
- [FREEZEHEADERoption](#)
- [FREEZELEFToption](#)
- [LABELoption](#)
- [Noption](#)
- [NODUPKEYoption](#)
- [NOGRANDTOTALoption](#)
- [NOOBSoption](#)
- [STYLEoption](#)
- [URLoption](#)

### Column Options:

- [ALIGNoption](#)
- [FORMAToption](#)
- [LABELoption](#)
- [LENGTHoption](#)
- [NAMEoption](#)
- [TYPEoption](#)
- [STYLEoption](#)
- [WIDTHoption](#)

### 4.6.1. Table options

Table options are data filters. They can be applied whenever a statement references a table. Table options can be applied to **output** or **input** tables. In general, selecting data at the point of input is more efficient than selecting them at the point of output. However, in some cases selecting data at the output point is more efficient.

Data filter	Description
DROP=	Selects the columns to be dropped from the table. Use a blank space to separate column names
KEEP=	Selects the columns to be kept in the table. Use a blank space to separate column names
RENAME=	Renames columns
WHERE(<condition>)	Filters the data to be kept. To write the <condition>, use available <a href="#">Functions - Operators - Values</a> .
IN=	<p>Allows you to flag the table, a row comes from. If used, it must be the First element on qualifying a table and then may be tested (If..THEN...ELSE controls.) so you can check where a row comes from and trigger corresponding processing.</p> <p>Use only with SET and MERGE statements.</p> <p>Refer to example: <a href="#">DATA - SET - IN</a></p>

- KEEP | DROP (mutually exclusive).
- To choose between KEEP or DROP, consider the number of variables to list.
- You can combine several table options.

#### 4.6.1.1. Using KEEP

... table (KEEP = col\_1 col\_n);

#### 4.6.1.2. Using DROP

... table (DROP = col\_1 col\_n);

#### 4.6.1.3. Using WHERE

... table (WHERE(<expression\_1> <operator> <expression\_2>));

#### 4.6.1.4. Using IN

... table\_1 (IN=flag\_1) table\_2 (IN=flag\_2);

If flag\_1 <operator> flag\_2 THEN

</ subordinate statements>;

OUTPUT;

END;

#### 4.6.1.5. Using RENAME

```
... table (RENAME = (old_col_1 = new_col_1 old_col_n = new_col_n));
```

#### 4.6.1.6. Combining data filters

```
... table (KEEP = col_1 col_n WHERE(<expression>) IN=flag);
```

### 4.6.2. ALIGN option

#### 4.6.2.1. Description

Column option; can be applied to any statement that identifies a column. ALIGN defines the alignment of the cell content of the column.

#### 4.6.2.2. Syntax

```
... col (ALIGN = </ option>);
```

#### 4.6.2.3. Options

Option
CENTER
LEFT
RIGHT

#### 4.6.2.4. Comment

- ALIGN options are predefined options.
- Columns with numerical values are by default right aligned.
- Columns with char values are by default left aligned.

#### 4.6.2.5. Error messages

Message	Explanation
Invalid ALIGN value	The value assigned to ALIGN is invalid

## 4.6.3. FORMAT option

### 4.6.3.1. Description

Column option; can be applied to any statement that identifies a column. Specifies the format to apply to the column(s) that precede the FORMAT option.

### 4.6.3.2. Syntax

... col (FORMAT = format.);

### 4.6.3.3. Comment

- format : [customized format](#) | EXCEL format.
- A period must follow the format.

### 4.6.3.4. Error messages

Message	Explanation
ERROR(0130)':' expected	The ':' is missing after the format name
<a href="#">Format not found</a>	Format couldn't be found.

## 4.6.4. FREEZEHEADER option

### 4.6.4.1. Description

Table option, that can be applied to the PROC PRINT statement, that opens the PROC PRINT step. Freezes the column headers, so that the user can scroll through the report without loosing the column headers. If printed, the column headers appear on each page.

### 4.6.4.2. Syntax

PROC PRINT DATA = table\_to\_print (</ [table\\_options](#)>) FREEZEHEADER;

RUN;

### 4.6.4.3. Comment

- We recommend you to use the FREEZELEFT option together with the FREEZEHEADER option.

## 4.6.5. FREEZELEFT option

### 4.6.5.1. Description

Table option, that can be applied to the PROC PRINT statement, that opens the PROC PRINT step. Freezes the columns on the left of a printable report. FREEZELEFT defines how many columns on the left are to be frozen.

### 4.6.5.2. Syntax

```
PROC PRINT DATA = table_to_print (</ table_options>) FREEZELEFT = n;
```

```
RUN;
```

### 4.6.5.3. Comment

- n: integer;
- We recommend you to use the FREEZELEFT option together with the FREEZEHEADER option.

## 4.6.6. LABEL option

### 4.6.6.1. Description

Valid as table option in steps : PROC PRINT, PROC TABULATE, PROC GCHART, PROC TRANSPOSE. Valid as column option, and therefore applicable to any statement that identifies a column. When used as table option, LABEL indicates that all columns are to be displayed using their label. When used as a variable option, LABEL assigns a label to the column.

### 4.6.6.2. Syntax - table option

```
... table_to_processLABEL;
```

### 4.6.6.3. Syntax - column option

```
... col (LABEL = "label");
```

### 4.6.6.4. Comment

- label : character string of up to 256 characters.

- By default, columns are displayed with the column name as column title. Use the LABEL option to customize column titles.
- Low-level label definitions always override high-level LABEL definitions.
- You can define bilingual labels using the following syntax:  
LABEL="Company{F}Société").

#### 4.6.6.5. Error messages

Message	Explanation
ERREUR(0121): {Chaîne de caractères} attendue	The value assigned to LABEL is not a text string

#### 4.6.7. LENGTH option

##### 4.6.7.1. Description

Column option; can be applied to any statement that identifies a column. LENGTH defines ...

##### 4.6.7.2. Syntax

... col (LENGTH = length);

##### 4.6.7.3. Comment

- length: Integer.

##### 4.6.7.4. Error messages

Message	Explanation
'INTEGER' expected after LENGTH	The value assigned to LENGTH must be an integer

#### 4.6.8. N option

##### 4.6.8.1. Description

Table option, that can be applied to the PROC PRINT statement, that opens the PROC PRINT step. Calculates the total of column values, and displays the totals.

#### 4.6.8.2. Syntax

```
PROC PRINT DATA = table_to_print (</ table_options>) N="Total";  
RUN;
```

#### 4.6.8.3. Comment

- N="Total": eFront Script calculates the total of table lines, and displays it preceded by the text string between quotes.
- If column values are regrouped using the BY statement, N calculates subtotals per group and the total of table lines: N="TotalGroup\_1" "TotalGroup\_n" "Total"; The last value to be calculated is the total number of table lines. The preceding values are group subtotals.
- N="": If no label is specified, eFront Script displays group subtotals and the total without a preceding label.
- N values and text can be styled.

#### 4.6.8.4. Error messages

Message	Explanation
---------	-------------

#### 4.6.9. NAME option

##### 4.6.9.1. Description

Valid as statement option in the [VARstatement](#) (when used in a PROC CONVERTCURR step). The NAME option defines the column to fill with the converted amounts.

##### 4.6.9.2. Syntax - column option

```
PROC CONVERTCURR DATA = table_to_process (</ table_options>);
```

[VAR](#)

```
var_1 (NAME=column_to_be_filled) var_n (NAME=column_to_be_filled);
```

RUN;

#### 4.6.9.3. Error messages

Message	Explanation

#### 4.6.10. NOGRANDTOTAL option

##### 4.6.10.1. Description

Table option, that can be applied to the PROC PRINT statement, that opens the PROC PRINT step.

##### 4.6.10.2. Syntax

```
PROC PRINT DATA = table_to_print (</ table_options>) NOGRANDTOTAL;  
RUN;
```

#### 4.6.11. NODUPKEY option

##### 4.6.11.1. Description

Table option, that can be applied to the PROC SORT statement, that opens the PROC SORT step. Eliminates duplicates in BY column values.

##### 4.6.11.2. Syntax

```
PROC SORT DATA = table_to_process (</ table_options>) OUT =  
output_table NODUPKEY;  
  
BY col_1(</ options>) col_n (</ options>);  
  
</ subordinate statements>;  
  
RUN;
```

##### 4.6.11.3. Comment

- NODUPKEY searches for duplicates only in BY column values.
- As soon as two values in one column are identical, one of the lines is eliminated.

- To keep the table\_to\_process intact, always combine NODUPKEY with the OUT option, and define an output\_table d, otherwise **eFront Script** eliminates duplicates from the table\_to\_process.

#### 4.6.11.4. Error messages

Message	Explanation
---------	-------------

### 4.6.12. NOOBS option

#### 4.6.12.1. Description

Table option, that can be applied to the PROC PRINT statement, that opens the PROC PRINT step. Removes the first column from the table to print (**eFront Script** generates automatically a first column in each output table that lists column line numbers).

#### 4.6.12.2. Syntax

PROC PRINT DATA = table\_to\_print (</ [table\\_options](#)>) NOOBS;

RUN;

#### 4.6.12.3. Comment

- To refer to the first column of a table, use the OBS variable.

#### 4.6.12.4. Error messages

Message	Explanation
---------	-------------

### 4.6.13. STYLE option

#### 4.6.13.1. Description

Valid as table option in steps: PROC TABULATE, PROC PRINT. Valid as statement option in the [TABLEstatement](#) and [VARstatement](#) (when used in a PROC PRINT step). The STYLE option defines either the [stylesheet](#) to be used for the output table, or it defines the style to apply to a particular element of the output table.

#### 4.6.13.2. Syntax - table option

Two ways of using STYLE as table option:

- 1: ... table\_to\_processSTYLE = stylesheet\_name;
- 2: ... table\_to\_processSTYLE(item\_to\_be\_styled) = (<style\_attributes>);

#### 4.6.13.3. Syntax - column option

STYLE as table option in a PROC TABULATE step:

```
PROC TABULATE DATA = table_to_process (</ table_options>;

CLASS col_1 col_n;

VAR var_1 var_n;

TABLE var_1 = (LABEL = "label_1" STYLE(item_to_be_styled = </ style_attributes>),
               </ column_structure>*
               </ functions>;

RUN;
```

STYLE as table option in a PROC PRINT step:

```
PROC PRINT DATA = table_to_print (</ table_options>);

VAR

var_1 (FORMAT=format.LABEL="label" STYLE=(item_to_be_styled= </
style_attributes>))

.function_name(VAR=colFORMAT=format.LABEL="label" STYLE(DATA)= </
style_attributes> STYLE(TOTAL) = </ style_attributes>);

RUN;
```

#### 4.6.13.4. Comment

- Function names within the VAR statement, must be preceded by a period.
- Low-level STYLE definitions always override high-level style definitions.

#### 4.6.13.5. Error messages

Message	Explanation
ERROR(0113): Stylesheet name expected	No stylesheet name has been assigned to the STYLE option when used as table option
ERROR: Invalid style element	The item to be styled may be invalid.
ERROR: Invalid style parameter	The style attribute may be incorrect or the item to be styled may be invalid

### 4.6.14. TYPE option

#### 4.6.14.1. Description

Column option; can be applied to any statement that identifies a column. TYPE defines the type of a column.

#### 4.6.14.2. Syntax

... col (TYPE = type);

#### 4.6.14.3. Comment

- type: List of predefined values.

#### 4.6.14.4. Types

Possible values	
BOOLEAN	INTEGER
BYTE	LOGICAL
CHAR	LONG
DATE	REAL
DATETIME	STRING
DOUBLE	TEXT

FLOAT	TIME
-------	------

- STRING and TEXT are synonyms

#### 4.6.14.5. Error messages

Message	Explanation
Invalid TYPE value	The value assigned to TYPE is invalid

#### 4.6.15. URL option

##### 4.6.15.1. Description

Table option, that can be applied to the PROC PRINT statement, that opens the PROC PRINT step. Specifies the link that allows to access the variable value from within a table, i.e. you create a clickable link that allows to access object instances.

##### 4.6.15.2. Syntax

PROC PRINT DATA = table\_to\_print (</ table\_options>)

URL = (TYPE="folder\_name"VAR=var)

;

RUN;

##### 4.6.15.3. Options

Option	Explanation
TYPE	Indicates the folder name  folder_name:  ex. : "TXT_REPORT.FOLDER=F_VCINESTINS"
VAR	Defines the column where clickable links are provided for each of the column values

## 4.6.16. WIDTH option

### 4.6.16.1. Description

Column option; can be applied to any statement that identifies a column. WIDTH defines the width of the column.

### 4.6.16.2. Syntax

... col (WIDTH = width);

### 4.6.16.3. Comment

- width: Integer.

### 4.6.16.4. Error messages

Message	Explanation
---------	-------------

## 4.7. Macro controls

Macro controls apply to programming steps, and macro statements. They allow to control the **flow of programming steps**, and **macro statements**. Here is the list of available controls:

- %DO WHILE...%LOOP WHILE
- %FOR...%NEXT
- %IF...%THEN...%END
- %SELECT... %WHEN... %THEN... %END
- %WHILE...%END

### 4.7.1. %DO WHILE...%LOOP WHILE

#### 4.7.1.1. Description

Macro control. Applies to programming steps, and macro statements. You can treat a group of programming steps or/and macro statements as a unit by putting them into a %DO loop. The %DO loop executes repetitively while a condition is true, checking the condition before each iteration of the %DO loop. The expression is evaluated at the top of the loop before the statements in the %DO loop are executed. If the expression is true, the %DO loop iterates. If the expression is false the first time it is evaluated, the %DO loop does not iterate even once.

#### 4.7.1.2. Syntax %DO WHILE

```
%DO WHILE <condition>
```

```
</ subordinate statements>;
```

```
%EXIT %DO;
```

```
%LOOP;
```

#### 4.7.1.3. Syntax %LOOP WHILE

```
%DO
```

```
</ subordinate statements>;
```

```
%EXIT %DO;
```

```
%LOOP WHILE <condition>;
```

#### 4.7.1.4. Comment

- <condition>: any **macro expression** that resolves to a logical value. The macro processor evaluates the expression at the top of each iteration. The expression is true if it is an integer other than zero. The expression is false if it has a value of zero.
- You can use %EXIT %DO to leave the %DO loop.
- You can nest %IF, %DO, and %FOR controls.

#### 4.7.2. %FOR...%NEXT

##### 4.7.2.1. Description

Macro control. The %FOR...%NEXT control runs repetitively while a condition is true, checking the condition before each iteration. The expression is evaluated at the top of the loop before the statements in the %FOR loop are executed. If the expression is true, the %FOR loop iterates. If the expression is false the first time it is evaluated, the %FOR loop does not iterate even once.

##### 4.7.2.2. Syntax

```
%FOR <condition> %TO <condition>
```

```
</ subordinate statements>;
```

```
%EXIT %FOR;
```

```
%NEXT;
```

##### 4.7.2.3. Comment

- <condition>: any **macro expression** that resolves to a logical value. The macro processor evaluates the expression at the top of each iteration. The expression is true if it is an integer other than zero. The expression is false if it has a value of zero.
- Use EXIT FOR to leave the loop if necessary.
- When using %FOR %I%TO, you don't necessarily need to declare %I beforehand, **eFront Script** creates it dynamically. Ex.: %FOR %I=1%TO 10.
- You can nest %IF, %DO, and %FOR loops.

- When using %FOR %I%IN (list\_of\_values), list\_of\_values can refer to a **macro variable!**

#### 4.7.2.4. Error messages

Message	Explanation
'TO' expected	The TO keyword is missing or misspelled.

#### 4.7.3. %IF...%THEN...%END

##### 4.7.3.1. Description

Macro control. The %IF expression is evaluated at the top of the loop before the statements in the %IF loop are executed. If the expression is false, the %IF control statements are not executed.

##### 4.7.3.2. Syntax

```
%IF <condition> %THEN
</ subordinate statements>;
%ELSEIF <condition> %THEN
</ subordinate statements>;
%ELSE
</ subordinate statements>;
%END;
```

##### 4.7.3.3. Comment

- <condition>: any **macro expression** that resolves to a logical value.
- At least, you need the keywords: %IF...%THEN...%END
- If you use the %ELSEIF statement, it must immediately follow the %IF...%THEN statement.
- If you use the %ELSE statement, it must immediately follow the %IF...%THEN statement or the %ELSEIF...%THEN statement.

- You can nest %IF, %FOR,%DO controls.
- Use a %SELECT control rather than a series of %IF...%THEN statements when you have a long series of mutually exclusive conditions.

#### 4.7.3.4. Error messages

Message	Explanation
ERROR(0113): 'THEN' expected	theN keyword may be missing or being misspelled. It could be too, that the error is due to an erroneous use of the conditional operator (AND, OR,...).

### 4.7.4. %SELECT... %WHEN... %THEN... %END

#### 4.7.4.1. Description

Macro control. %SELECT...%WHEN...%THEN...%END controls allow you to execute an important number of conditional groups of statements. The %SELECT keyword evaluates the variable value to consider, whereas the %WHEN...%THEN statements list the groups of statements to execute according to the value taken by the variable.

#### 4.7.4.2. Syntax

%SELECT %var

%WHEN (var\_value\_1) %THEN </ subordinate statements>;

%END;

...

%WHEN (<var\_value\_2>) %THEN </ subordinate statements>;

%END;

%OTHERWISE </ subordinate statements>;

%END;

%END;

#### 4.7.4.3. Comment

- select\_value is an expression that evaluates to a single value.
- when\_value is a constant or an expression that evaluates to a single value.
- Use at least one WHEN...THEN statement.
- Note that each WHEN...THEN statement must be followed by an END; statement.
- Use the OTHERWISE statement to define the action to take, when all the WHEN conditions are false. Yet, the OTHERWISE statement is not required.
- If you don't specify a statement after THEN, **eFront Script** recognizes the WHEN...THEN statement as true.
- Use IF...THEN...ELSE controls for programs with few statements, and SELECT controls for programs with an important number of conditional statements.

#### 4.7.4.4. Error messages

Message	Explanation
ERROR: WHEN, OTHERWISE or END expected	The keywords WHEN, OTHERWISE or END are expected. The keyword could be missing or being misspelled.

### 4.7.5. %WHILE...%END

#### 4.7.5.1. Description

Macro control. Executes a group of statements repetitively while a condition is true. The expression is evaluated at the top of the loop before the statements in the %WHILE loop are executed. If the expression is true, the %WHILE loop iterates. If the expression is false the first time it is evaluated, the %WHILE loop does not iterate even once.

#### 4.7.5.2. Syntax

```
%WHILE <condition>  
</ subordinate statements>;  
%END;
```

#### 4.7.5.3. Comment

- <condition>: any macro expression that resolves to a logical value. The macro processor evaluates the expression at the top of each iteration. The expression is true if it is an integer other than zero. The expression is false if it has a value of zero.

#### 4.7.6. Create a macro loop counter

To create a macro loop counter, prefix the counter using %.

Here is an example:

```
%FOR  
%FOR%I=1 %TO 12  
  
DATA SET ...;  
  
RUN;  
  
PROC PRINT ...;  
  
RUN;  
  
%NEXT ;
```

## 4.8. Step controls

**Step controls** only occur **within** a programming step. For the moment they only are valid in **DATA** steps. Controls allow to control the **flow of statements**. Here is the list of available controls:

- **DO WHILE...DO LOOP**
- **FOR...TO...NEXT**
- **FOR...IN...NEXT**
- **IF...THEN...END**
- **SELECT...WHEN...THEN...END**

### 4.8.1. DO WHILE...LOOP WHILE

#### 4.8.1.1. Description

Valid in DATA steps. You can treat a group of statements as a unit by putting them into a DO loop. The DO loop executes repetitively while a condition is true, checking the condition before each iteration of the DO loop. The expression is evaluated at the top of the loop before the statements in the DO loop are executed. If the expression is true, the DO loop iterates. If the expression is false the first time it is evaluated, the DO loop does not iterate even once.

#### 4.8.1.2. Syntax DO WHILE

DATAoutput\_table (</ table\_options>);

...;

DO WHILE <condition>

</ subordinate statements>;

EXIT DO;

LOOP;

...;

RUN;

#### 4.8.1.3. Syntax LOOP WHILE

```
DATAoutput_table (</ table_options>);
```

```
...;
```

```
DO
```

```
</ subordinate statements>;
```

```
EXIT DO;
```

```
LOOP WHILE <condition>;
```

```
...;
```

```
RUN;
```

#### 4.8.1.4. Comment

- You can use EXIT DO to leave the DO loop.
- You can nest IF, FOR and DO loops.

#### 4.8.1.5. Error messages

Message	Explanation
---------	-------------

### 4.8.2. FOR...TO...NEXT

#### 4.8.2.1. Description

Valid in DATA steps. The FOR...TO...NEXT control runs repetitively while a condition is true, checking the condition before each iteration. The expression is evaluated at the top of the loop before the statements in the FOR loop are executed. If the expression is true, the FOR loop iterates. If the expression is false the first time it is evaluated, the FOR loop does not iterate even once.

#### 4.8.2.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
...;  
FORcol=<expression> TO <expression>  
</ subordinate statements>;  
EXIT FOR;  
NEXT;  
...  
RUN;
```

#### 4.8.2.3. Comment

- Use EXIT FOR to leave the loop if necessary.
- You can nest IF, DO, and FOR loops.

#### 4.8.2.4. Error messages

Message	Explanation
'TO' expected	The TO keyword is missing or misspelled.

### 4.8.3. FOR...IN...NEXT

#### 4.8.3.1. Description

Valid in DATA steps. The FOR...IN...NEXT control runs repetitively while values are part of a list being specified. The condition is checked before each iteration. The expression is evaluated at the top of the loop before the statements in the FOR loop are executed. If the expression is true, the FOR loop iterates. If the expression is false the first time it is evaluated, the FOR loop does not iterate even once.

#### 4.8.3.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
...;
```

```
FORvarINlist
```

```
</ subordinate statements>;
```

```
EXIT FOR;
```

```
NEXT;
```

```
...
```

```
RUN;
```

#### 4.8.3.3. Comment

- list can refer to a macro variable.
- Use EXIT FOR to leave the loop if necessary.
- You can nest IF, DO, and FOR loops.

#### 4.8.3.4. Error messages

Message	Explanation
'TO' expected	The TO keyword is missing or misspelled.

### 4.8.4. IF...THEN...END

#### 4.8.4.1. Description

Valid in: DATA steps. Executes statements that meet specific conditions. The IF expression is evaluated at the top of the loop before the statements in the IF loop are executed. If the expression is false, the IF control statements are not executed.

#### 4.8.4.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
...;
```

```
IF <condition> THEN
```

```
</ subordinate statements>;
```

```
ELSEIF <condition> THEN
```

```
</ subordinate statements>;  
ELSE  
</ subordinate statements>;  
END;  
...;  
RUN;
```

#### 4.8.4.3. Comment

- If you use the ELSEIF statement, it must immediately follow the IF...THEN statement.
- If you use the ELSE statement, it must immediately follow the IF...THEN statement or the ELSEIF...THEN statement.
- You can nest FOR, IF, and DO loops.
- Use a SELECT control rather than a series of IF...THEN statements when you have a long series of mutually exclusive conditions.

#### 4.8.4.4. Error messages

Message	Explanation
ERROR(0113): 'THEN' expected	theN keyword may be missing or being misspelled. It could be too, that the error is due to an erroneous use of the conditional operator (AND, OR,...).

### 4.8.5. SELECT...WHEN...THEN...END

#### 4.8.5.1. Description

Valid in: DATA steps. SELECT...WHEN...THEN...END groups allow you to execute an important number of conditional groups of statements. The SELECT keyword evaluates the variable value to consider, whereas the WHEN...THEN statements list the groups of statements to execute according to the value taken by the variable.

#### 4.8.5.2. Syntax

```
DATAoutput_table (</ table_options>);
```

...;

SELECT (select\_value)

WHEN (when\_value\_1) THEN </ subordinate statements>;

END;

...

WHEN (<when\_value\_2>) THEN </ subordinate statements>;

END;

OTHERWISE </ subordinate statements>;

END;

END;

...;

RUN;

#### 4.8.5.3. Comment

- select\_value is an expression that evaluates to a single value.
- when\_value is a constant or an expression that evaluates to a single value.
- Use at least one WHEN...THEN statement.
- Note that each WHEN...THEN statement must be followed by an END; statement.
- Use the OTHERWISE statement to define the action to take, when all the WHEN conditions are false. Yet, the OTHERWISE statement is not required.
- If you don't specify a statement after THEN, **eFront Script** recognizes the WHEN...THEN statement as true.
- Use IF...THEN...ELSE controls for programs with few statements, and SELECT controls for programs with an important number of conditional statements.

#### 4.8.5.4. Error messages

Message	Explanation
ERROR: WHEN, OTHERWISE or END expected	The keywords WHEN, OTHERWISE or END are expected. The keyword could be missing or being misspelled.

#### 4.8.6. Create a loop counter

To create a loop counter, declare the counter as you would declare a table **column**, initialize it, and refer to it.

Here is an example :

```
DATAoutput_table (</ table_options>);

COLUMN X TYPE=INTEGER;

X=0;

DO WHILE (X<5)

</ subordinate statements;

X=X+1

LOOP;

RUN;
```

## 4.9. Functions - Operators - Values

Available functions process:

- [Numbers](#)
- [Character strings](#)
- [Dates](#)
- [Functions - files and folders](#)

Find also special functions:

- [Random/lag/parent...](#)

And:

- [Operators](#)

And:

- [Special values](#)
- [Specifying values or value ranges](#)

### 4.9.1. Functions - numbers

Function	Description	Example
ABS(<number>)	Returns the absolute value of <number>. <number> = double	
ACOS(<number>)	Returns the inverse cosine (arccosine) of <number>	
ASIN(<number>)	Returns the inverse sine (arcsine) of <number>	
ATAN(<number>)	Returns the inverse tangent (arctangent) of <number>	
CBOOL(<expression>)	Returns a boolean value for an <expression>	
CDATE(<expression>)	Converts an expression into a date.	
CDBL(<expression>)	Converts an <expression> to a double.	

CEILING(<number>)	Returns the smallest integer not less than <number>. <number> = double	refer to example: <a href="#">WEEKDAY()</a>
CINT(<expression>)	Converts an expression to the nearest integer.	refer to example: <a href="#">ROUND()_CINT()</a>
COS(<number>)	Returns the cosine of <number>	
COSH(<number>)	Returns the hyperbolic cosine of <number>	
CSTR(<expression>)	Converts an <expression> into a character string.	
DISTRIBNORM (<mean>, <stdev>)	Returns a random number normally distributed.  ⚠ Not available for eFront Cube programs.	PROC PRINT;  PUT DISTRIBNORM(10,5);  RUN;
DISTRIBTRIANG(<min>, <max>, <median>)	Returns a random number on a triangle distribution.  ⚠ Not available for eFront Cube programs.	PROC PRINT;  PUT DISTRIBTRIANG(0,10,5);  RUN;
E	Returns the exponential value for base=1	
EXP(<number>)	Returns the exponential value for base=<number>	
FLOOR(<number>)	Returns the largest integer less than or equal to <number>. <number> = double	
LOG(<number>)	Returns the logarithm for base=<number>	
LOG10	Returns the logarithm for base=10	
MAX(<number-1> ... <number-n>)	Returns the maximum value amidst the numbers being specified.	
MIN(<number-1> ... <number-n>)	Returns the minimum value amidst the numbers being specified.	

N0(<number-1>,<number-2>)	Returns 0 if the absolute value of <number-1> is inferior to <number-2>	Example:  If NB_SHARES=2, then:  N0(NB_SHARES,0.001) returns 2.  If NB_SHARES=0.00001, then:  N0(NB_SHARES,0.001) returns 0.
PI	Returns the value for PI (prime counting function).	
POW(<number-1>, <number-2>)	Returns the value of <number-1> power <number-2>	
RANDOM(integer)	Returns a random number from the range between 0 and <integer>	PROC PRINT;  PUT RANDOM(10);  RUN;
ROUND(<value>, <remaining_digits>)	Rounds the <value> to the nearest integer.  <remaining_digits>=0: rounds the value to the nearest integer, no digits  <remaining_digits>=1: rounds the value to the nearest integer, 1 digital  <remaining_digits>=n: rounds the value to the nearest integer, n digital	refer to example:  <a href="#">ROUND()_CINT()</a>
ROUNDINGMETHOD(<value>, <remaining_digits>, "<method>")	Round the <value> with precision given by <remaining_digits>, by applying a specific method :  --> "LOWER"  --> "UPPER"  --> "NEAREST"	
RND()	Returns a random number between 0 and 1.	
SIN(<number>)	Returns the sine of <number>	
SINH(<number>)	Returns the hyperbolic sine of <number>	

SQRT(<number>)	Returns the square root of <number>	refer to example:  PROC MEANS - MEAN - SUM - SQRT()
TAN(<number>)	Returns the tangent of <number>	
TANH(<number>)	Returns the hyperbolic tangent of <number>	
TOBOOL(<expression>)	Returns a boolean value for an <expression>	
TODAT(<expression>)	Converts an expression into a date.	
TODBL(<expression>)	Converts an <expression> to a double.	
TOINT(<expression>)	Converts an expression to an integer.	
TOSTR(<expression>)	Converts an <expression> into a character string.	

## 4.9.2. Functions - character strings

Function	Description	Example
ASC(<char>)	Returns the ASCII code of a character	
CHR(<UNICODE_code>)	Converts <UNICODE_code> to <char>	STR1 = 'Oliver'; STR2 = 'Michel'; STR = STR1 + CHR(11) + STR2; Result: Oliver Michel Comment: CHR(11) allows you to concatenate two strings in a cell a start each string in a new line. You need to insert a a VT (Vertical tabulation).
COMPRESS(<string>)	Removes all blanks from a <string>.	A=' Robert and Mary '; B=COMPRESS(A); Result : 'RobertandMary';

CRLF()	Returns WINDOWS compatible new line character (CRLR).  This may be useful if concatenating strings.	STR1 = 'Oliver';  STR2 = 'Michel';  STR = STR1 + CRLF() + STR2;  Result:  Oliver Michel
CSTR<expression>	Converts an <expression> into a character string	
FIND(<string-1>, <string-2>)	Returns the position of <string-2> in <string-1>, positive integer.  If <string-2> is not part of <string-1>, the result = '0'.  Comparison starts from the left.	A='ROBERT';  B=FIND(A,'B');  Result: 3
FORMAT(<string>, <format>)	Applies the <format> to the <string>	'd/M/yy'
INSTR(<string-1>, <string-2>)	Returns the start position of the <string-2> within <string-1>, comparison starts from the left. Returns 0, if search is unsuccessful	A='ROBERT';  B=INSTR(A,'BERT');  Result: 3
LANG("<string>{f}<Frenchstring>")	Localizes the input <string> based on the application current language settings	LANG("Fund{F}Fonds") returns Fund if the application language is set to English, Fonds if it is set to French.
LCASE(<string>)	Converts all letters in a <string> to lowercase	A='ROBERT';  B=LOWCASE(A);  Result: 'robert'
LEFT(<string>,<length>)	Returns the characters on the left of a <string>, with the returned string's length = <length>	A='Robert';  B=Left(A,2);  Result: 'Ro'
LEN(<string>)	Returns the length of the <string>	A='Robert';  B=LEN(A);  Result: 6

LENGTH(<string>)	Returns the length of the <string>	A='Robert'; B=LENGTH(A); Result: 6
LOWCASE(<string>)	Converts all letters in a <string> to lowercase	A='ROBERT'; B=LOWCASE(A); Result: 'robert'
LTRIM(<string>)	Removes all trailing blanks on the left, and returns the trimmed argument as a result. Blanks within the string are replaced by a single blank.	A=LTRIM(' Robert'); Result: "Robert"
MID(<string>, <position>, <length>)	Returns a substring of a <string>. The substring is identified by its start position and length.	A='Robert'; B=MID(A,2,4); Result:'ober'
REPLACE(<string>, <string_to_replace>, <new_string>)	Replaces in a <string>, the <string_to_replace> by a <new_string>	A='movement'; B=REPLACE(A,"e","i"); Result:'mivimint'
RIGHT(<string>, <length>)	Returns the characters on the right of a <string>, with the returned string's length = <length>	A='Robert'; B=RIGHT(A,4); Result: 'bert'
RTRIM(<string>)	Removes all trailing blanks on the right, and returns the trimmed argument as a result. Blanks within the <string> are replaced by a single blank	A=RTRIM('Robert '); Result: "Robert"
SPLIT(<string>, <separator>)	Creates a collection of items. Comma is the default separator.	PROC PRINT; PUT SPLIT("a,b,c,d"); RUN;
STRIP(<string>)	Returns a character string with all leading and trailing blanks removed	
SUBSTR(<variable>, <position>, <length>) =<string>	Returns TRUE   FALSE. SUBSTR compares <string> to the substring of the variable identified by its start position and length.	A='Robert'; B=(SUBSTR(A,1,2)='Ro'); Result : True

TRIM(<string>)	Removes all trailing blanks, and returns the trimmed argument as a result. Blanks within the string are replaced by a single blank. TRIM is useful for concatenating because concatenation does not remove trailing blanks. TRIM (X) = LTRIM(RTRIM(X))	A=TRIM("Robert and Mary");  Result: "Robert and Mary"
UCASE(<string>)	Converts all letters in a string to uppercase	A='robert';  B=UCASE(A);  Result: 'ROBERT'
UPCASE(<string>)	Converts all letters in a string to uppercase	A='robert';  B=UPCASE(A);  Result: 'ROBERT'
GETURLWEBSITE ()	Returns the URL of the current website	%LET MYVAR = GETURLWEBSITE();  PROC PRINT;  PUT %MYVAR;  RUN;

#### 4.9.3. Functions - dates

Function	Description	Examples
CDATE(<expression>)	Converts an expression into a date.	Refer to example:  <a href="#">WEEKDAY()</a>

DATEADD("<period>",<nb>,<date>)	<p>Adds &lt;nb&gt; times the &lt;period&gt; to a specified &lt;date&gt;.</p> <p>&lt;period&gt;: DAY   WEEK   MONTH   QUARTER   YEAR</p> <p>&lt;nb&gt;: integer (&lt;nb&gt; can be a negative value, and the period is subtracted &lt;nb&gt; times from the &lt;date&gt;)</p> <p>&lt;date&gt;: date value</p>	<p>Refer to examples:</p> <p><a href="#">DATEADD() - FORMAT()</a></p> <p><a href="#">DATEDIFF() - SEMESTERENDDATE() - 2</a></p> <p><a href="#">DATEDIFF() - QuarterBegDate() - DateADD()</a></p> <p>Another example:</p> <p>TEST_DATE = (DATEADD("WEEK,-1,%EDATE));</p> <p>Subtracts One week from date %EDATE</p>
---------------------------------	--	--

DATEDIFF("<period>",<date-1>,<date-2>)	<p>Returns the difference between &lt;date-1&gt; and &lt;date-2&gt;, expressed in &lt;period&gt;.</p> <p>&lt;period&gt;: DAY   WEEK   MONTH   QUARTER   YEAR   SEMESTER*</p> <p>*defines a period of 6 months</p> <p>if &lt;date-1&gt; &lt; &lt;date-2&gt;, the result is positive</p> <p>if &lt;date-1&gt; &gt; &lt;date-2&gt;, the result is negative</p>	<p>Refer to examples:</p> <p><a href="#">DATEDIFF() - FORMAT()</a></p> <p><a href="#">DATEDIFF() - SEMESTERENDDATE() - 1</a></p> <p><a href="#">DATEDIFF() - SEMESTERENDDATE() - 2</a></p> <p><a href="#">DATEDIFF() - QuarterBegDate() - DateADD()</a></p>
DATESERIAL(<year>, <month>, <day>)	Returns the date corresponding to <year>, <month> and <day>	<pre>PROC PRINT; PUT FORMAT(DATESERIAL(2006,5,13),"d/M/yyyy"); RUN;</pre> <p>Note that you need to use an upper-case m (M) for the month. If a lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p> <p>Result: 13/5/2006</p>
DAY(<date>)	Returns the day of the date value.	<p>Refer to example:</p> <p><a href="#">DMY() - MDY() - FORMAT()</a></p>

DAYOFYEAR(<date>)	Returns the number of the day, the first january being the number 1.	<pre>PROC PRINT; PUT DAY("18/07/2006"); RUN;</pre> <p>Result: 199</p>
DML(<date>)	Puts the date in the right format for the database (SQL Server).	<pre>PROC PRINT; PUT DML("31/12/2016"); RUN;</pre>
DMY(<day>,<month>,<year>)	Returns the day, the month, and the year from a date value.	<p>Refer to example:</p> <p><a href="#">DMY() - MDY() - FORMAT()</a></p>
FORMAT(<date>, "<format>")	<p>Applies a format to a date.</p> <p>&lt;date&gt;: a date or a column whose type is being set to DATE</p> <p>&lt;format&gt;: date picture, example: "d/M/yyyy" "dd/MM/yyyy" or "M/d/yyyy"</p>	<p>Refer to example:</p> <p><a href="#">YMD() - FORMAT()</a></p> <p><a href="#">DMY() - MDY() - FORMAT()</a></p> <p><a href="#">PROC EXPORT - FORMAT()</a></p>
MDY(<date>)	Returns the month, the day, and the year from a date value.	<p>Refer to example:</p> <p><a href="#">DMY() - MDY() - FORMAT()</a></p>

MONTH(<date>)	Returns the month from a date value.  If used together with the CLASS statement in a PROC MEANS step, <date> is set to the last day of the month.	Refer to example:  <a href="#">DMY() - MDY() - FORMAT()</a>
MONTHBEGDATE(<date>)	Returns the first day of the month being specified in the date value.	<pre>PROC PRINT; PUT FORMAT(MONTHBEGDATE(Today()), "d/M/yyyy"); RUN;</pre> <p>Note that you need to use an upper-case m (M) for the month. If a lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p> <p>Result: the first day of the current month</p>
MONTHENDDATE(<date>)	Returns the last day of the month being specified in the date value.	<pre>PROC PRINT; PUT FORMAT(MONTHENDDATE(Today()), "d/M/yyyy"); RUN;</pre> <p>Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p> <p>Result: the last day of the current month</p>
NOW()	Returns the current date and time.	<p>Refer to example:</p> <p><a href="#">PROC SQLTABLE - (2)</a></p>

QUARTER(<date>)	Returns the quarter that the date is in.  If used together with the CLASS statement in a PROC MEANS step, <date> is set to the last day of the quarter.	PROC PRINT;  PUT QUARTER("19/08/2016");  RUN;  Result: 3  Refer to example:  <a href="#">DATEDIFF() - QuarterBegDate() - DateADD()</a>
QUARTERBEGDATE(<date>)	Returns the first day of the quarter being specified in the date value.	PROC PRINT;  PUT FORMAT(QUARTERBEGDATE(TODAY()), "d/M/yyyy");  RUN;  Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.  Result: the first day of the current quarter  Refer to example:  <a href="#">DATEDIFF() - QuarterBegDate() - DateADD()</a>
QUARTERENDDATE(<date>)	Returns the last day of the quarter being specified in the date value.	PROC PRINT;  PUT FORMAT(QUARTERENDDATE(TODAY()), "d/M/yyyy");  RUN;  Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.  Result: the last day of the current quarter

SEMESTER(<date>)	Returns the semester that the date is in.  If used together with the CLASS statement in a PROC MEANS step, <date> is set to the last day of the semester.	PROC PRINT;  PUT SEMESTER ("19/08/2017");  RUN;  Result: 2  Refer to example:  <a href="#">PROC MEANS - SUM - SEMESTER()</a>
SEMESTERBEGDATE(<date>)	Returns the first day of the quarter being specified in the date value.	PROC PRINT;  PUT FORMAT(SEMESTERBEGDATE(TODAY()),"dd/MM/yyyy");  RUN;  Result: the first day of the current semester. Ex.: current date = 24/01/2007; semester start date = 01/01/2008  Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.
SEMESTERENDDATE(<date>)	Returns the last day of the semester being specified in the date value.	PROC PRINT;  PUT FORMAT(SEMESTERENDDATE(TODAY()),"dd/MM/yyyy");  RUN;  Result: the last day of the current semester. Ex.: current date = 24/01/2007; semester end date = 30/06/2007  Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.  Refer to example:  <a href="#">DATEDIFF() - SEMESTERENDDATE()_1</a>

TICKCOUNT()	Returns the number of milliseconds since the system started.	<pre>PROC PRINT; PUT TICKCOUNT(); RUN;</pre> <p>Result: 130209282</p>
TODAY()	Returns the current date as a date value. (TODAY is another name for the DATE function)	<pre>PROC PRINT; PUT FORMAT(TODAY(), "d/M/yyyy"); RUN;</pre> <p>Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p> <p>Result: the day of today</p>
WEEKBEGDATE(<date>)	Returns the first day (Monday) of the week being specified in the date value.	<pre>PROC PRINT; PUT FORMAT(WEEKBEGDATE(TODAY()), "d/M/yyyy"); RUN;</pre> <p>Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p> <p>Result: the date of Monday of the current week</p>
WEEKDAY(<date>)	Returns the number of the day from within a week (from a date value).	<pre>PROC PRINT; PUT WEEKDAY(TODAY()); RUN;</pre> <p>Result: returns the number of the current day within the week (1:monday; 2:tuesday; 3:wednesday;...)</p> <p>Refer to example: <a href="#">WEEKDAY()</a></p>

<b>WEEKENDDATE(&lt;date&gt;)</b>	Returns the last day (Saturday) of the week being specified in the date value.	<pre>PROC PRINT; PUT FORMAT(WEEKENDDATE(TODAY()), "d/M/yyyy"); RUN;</pre> <p>Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p> <p>Result: the date of Saturday of the current week</p>
<b>WEEKOFYEAR(&lt;date&gt;)</b>	Returns the number of the week, the first week of january being the week 1.	<pre>PROC PRINT; PUT WEEKOFYEAR(TODAY()); RUN;</pre> <p>Result: the number of the current week within the year</p>
<b>YEAR(&lt;date&gt;)</b>	<p>Returns the year from a date value.</p> <p>If used together with the CLASS statement in a PROC MEANS step, &lt;date&gt; is set to the last day of the Year.</p>	<p>Refer to example:</p> <p><a href="#">DMY() - MDY() - FORMAT()</a></p> <p><a href="#">DATEDIFF() - QuarterBegDate() - DateADD()</a></p>
<b>YEARBEGDATE</b>	Returns the first day of the year from a specified value.	<pre>PROC PRINT; PUT FORMAT(YEARBEGDATE(TODAY()), "d/M/ yyyy"); RUN;</pre> <p>Result: returns the date of the first January of the current year</p> <p>Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p>

YEARENDDATE	Returns the last day of the year from a specified value.	<pre>PROC PRINT; PUT FORMAT(YEARENDDATE(TODAY()),"d/M/ YYYY"); RUN;</pre> <p>Result: returns the date of the 31 December of the current year.</p> <p>Note that you need to use upper-case m (M) for the month. If lower-case m is used, the program will return '0' for the month as the lower-case m represents minutes.</p>
YMD(<year>, <month>, <day>)	Returns the year, the month, and the day from a date value.	<p>refer to examples:</p> <p><a href="#">YMD()</a></p> <p><a href="#">YMD() - FORMAT()</a></p>

#### 4.9.4. Functions - files and folders

Function	Description	Example
DIRECTORIES(<path>)	<p>Returns the number of directories being associated with the &lt;path&gt;. Root="\"</p> <p>DIRECTORIES() supports the following magic variables:</p> <ul style="list-style-type: none"> <li>--&gt; {SHARED}</li> <li>--&gt; {PUBLIC}</li> <li>--&gt; {PRIVATE}</li> <li>--&gt; {INFO}</li> </ul>	

DIRECTORY(<n>)	Returns the complete path to the directory at position <n>. <n>: integer, comprised between 1 and DIRECTORIES(). DIRECTOR() sends the directory at position <n> into a temporary folder, and makes it accessible.	
DIRECTORYNAME(<n>)	Returns the name of the directory at position <n>. <n>: userid, comprised between 1 and DIRECTORIES(). The directory is not extracted, and therefore it is not accessible.	
FILES(<path>)	Returns the number of files being associated with the <path>. Root="\". Does not take into account underlying folders!  FILES() supports the following magic variables: --> {SHARED} --> {PUBLIC} --> {PRIVATE} --> {INFO}	FILES("\{SHARED\}LOGOS") --> interpreted as: "\SHARED Region(company\LOGOS"
FILE(<n>)	Returns the complete path to the file at position <n>. FILE() sends the file at position <n> into a temporary folder, and makes it accessible. <n>: integer, comprised between 1 and FILES().	FOR I=1 TO FILES("\FRONTVENTURE (Private)\Company\Files")  FICHIER = FILE(I);  OUTPUT;  NEXT;  RUN;  Refer to example:  <a href="#">FILENAME() - FILE()</a>

<b>FILENAME(&lt;n&gt;)</b>	Returns the name of the file at position <n>. The file is not extracted, and therefore the file is not accessible. Use FILENAME(), if the file name only is important, FILENAME() is quicker than FILE()	Refer to examples:  <a href="#">FILENAME() - FILE()</a>  <a href="#">LINKFILES() - LINKFILE() - FILENAME()</a>
<b>FILESEXT(&lt;path&gt;, &lt;extension&gt;)</b>	Retrieves the number of files in <path>, in specified <extension>	Example:  PROC PRINT;  PUT FILESEXT("\SHARED", ".pdf");  RUN;  Returns: 1
<b>LINKFILES(&lt;object_id&gt;)</b>	Returns the number of files that are attached to the object identified by the <object_id>	Refer to examples:  <a href="#">LINKFILES() - LINKFILE() - FILENAME()</a>  <a href="#">LINKFILES() - LINKFILE() - FILENAME() - %PARAM</a>  <a href="#">%LET - LINKFILES() - LINKFILE()</a>
<b>LINKFILESEXT(&lt;object_id&gt;, &lt;ext_1&gt;; &lt;ext_n&gt;)</b>	Returns the number of files that are attached to the object identified by the <object_id>. Files are filtered according to the list of file extensions.	Refer to examples:  <a href="#">LINKFILES() - LINKFILE() - FILENAME()</a>  For I=1 TO  LINKFILESEXT(COMPANY_ID, ".PNG")  LOGO = LINKFILE(I);  OUTPUT;  NEXT;
<b>LINKFILE(&lt;n&gt;)</b>	Returns the complete path that leads to file <n> of the files attached to an object. <n>: integer, comprised between 1 and LINKFILES(). LINKFILE() copies the file into a temporary folder, and makes it accessible.	use this function to retrieve and use images  Refer to example:  <a href="#">LINKFILES() - LINKFILE() - FILENAME()</a>

LINKFILENAME(<n>)	Returns the name of file <n> of the files attached to an object. <n>: integer, comprised between 1 and LINKFILES(). When using LINKFILENAME(), the file itself is not accessible.	
MAKEDIRECTORY(<dir>)	Creates a subdirectory in the current directory, which you must define (e.g. LIBNAME USER CURRENT)	<p>Example:</p> <pre>LIBNAME USER CURRENT; PROC PRINT; PUT MAKEDIRECTORY("MyNewSubDir"); RUN;</pre> <p>Returns: True. The directory is created.</p>

#### 4.9.5. Functions - special

Function	Description	Example
APPENDPDF("<file_name1>","<file_name2>")	<p>Appends the content of an existing PDF file to another PDF file.</p> <p>Note that both PDF files must be located on the application server. The program will not compile if one or both files are attached to the program or stored elsewhere.</p> <p> Not available for eFront Cube programs</p>	<pre>%LET MERGED_FILE=APPENDPDF("D:\eFront\download\Document.pdf", "D:\eFront\download\Document2.pdf");</pre> <pre>PROC PRINT; PUT %MERGED_FILE; RUN;</pre> <p>Result:</p> <p>All pages from Document2.pdf are appended to Document.pdf.</p>
CURRLITERAL(<value>, <language>, <currency>, 'centime')	<p>Converts a &lt;value&gt; into string. &lt;language&gt;: FRENCH   ENGLISH</p> <p>&lt;currency&gt;: EURO, FRANC, LIVRE, DOLLAR</p>	<pre>PUT CURRLITERAL (12.2, 'FRENCH', 'euro', 'cent');</pre> <p>Result: douze euros et vingt cents</p>

CONVERTCURR(<amount>, <source_curr>, <target_curr>, <date>)	<p>Converts an &lt;amount&gt; from the &lt;source_curr&gt; currency into the &lt;target_curr&gt; at the rate of a given &lt;date&gt;</p> <p>If you reference an empty parameter it is critical that you use " " instead of "" (see example)</p>	<p>Example:</p> <pre>RATE = CONVERTCURR (1, FUND_CURRENCY, INVESTOR_CURRENCY, REFERENCE_DATE);  FX = CONVERTCURR2(1,INVESTEE_FUND_CCY, MANAGED_FUND_CCY,ENTRY_DATE, ",CURRENCY_TABLE_IQID);</pre>
CONVERTCURR2(<amount>, <source_curr>, <target_curr>, <date>, "<table_name>", "<table_id>")	Converts an <amount> from the <source_curr> currency into the <target_curr> at the rate of a given <date>, using a specific conversion table (given by its id <table_id>, or its name <table_name>)	
COLLECTION("<table_name>", "<Column name>")	Returns a collection containing all values of the specified Table/Column	<p>Example:</p> <pre>%LET LIST_ID = COLLECTION ("FV_T_INFO_FUND", "FUND_ID");</pre>
EXISTCOLUMN("<table_name>", "<column_name>")	Tests the existence of a column in a table	
EXISTMACRO("<macro_name>")	Tests the existence of a macro	<p>Refer to example:</p> <pre>%DEFINE</pre>
EXISTTABLE("<table_name>")	Tests the existence of a table	
GETTEMPPATH()	<p>Returns the complete path of the user's download on the webserver.</p> <p>Ex.:GETACCCO 'C:\inetpub\wwwroot\website\download\ 93930c959a734c8bb41f1324 e23d244a'</p>	<p>Refer to example:</p> <pre>GETTEMPPATH() PROC EXPORTEXCEL - GETTEMPPATH()</pre>
GETACCOUNTID()	Returns the unique identification number of the current account	

GETCONTACTUSERID()	Returns the ID of the specified contact.  ⚠ Only applies to eFront Cube programs	Example:  PROC PRINT;  PUT GETDEFAULTCONTACTID();  RUN;
GETCURRENTUSERID()	Returns the ID of the current user  ⚠ Only applies to eFront Cube programs	Example:  PROC PRINT;  PUT GETCURRENTUSERID();  RUN;  Returns: D7CD41F68C6F456399A66ED3D0BA906B
GETCURRENTUSERINFO("<fieldname>")	Returns the user specified <fieldname> value	Example:  PROC PRINT;  PUT GETCURRENTUSERINFO("FIRSTNAME1");  RUN;  Returns: Olivier
GETCURRENTUSERPROFILE()	Returns the user's current profile ID  ⚠ Only applies to eFront Cube programs	Example:  %LET MyProfileName = GETCURRENTUSERPROFILE();  TRACE "My current profile ID is: " & %MyProfileName;
GETCURRENTUSERPROFILESLIST()	Not authored yet  ⚠ Only applies to eFront Cube programs	Not authored yet
GETUSERPROFILESLIST ("<user_id>")	Returns the list of unique identifiers of all the user's profile (separated by ";")	Example:  %LET MyProfiles = GETUSERPROFILESLIST(GetUserID());  TRACE %MyProfiles;

GETDBSERVER()	Returns the DB server type	<p>Example:</p> <pre>PROC PRINT; PUT GETDBSERVER(); RUN;</pre> <p>Returns: MSSQL</p> <p>Other possible values: "ACCESS", "MSSQL"</p>
GETDEFAULTUSERID()	Returns the default user ID of the current user.  ⚠ Not available for eFront Cube programs	<p>Example:</p> <pre>PROC PRINT; PUT GETDEFAULTUSERID(); RUN;</pre>
GETINFOREGIONID()	Returns the ID of the INFO region.	<p>Example:</p> <pre>PROC PRINT; PUT GETINFOREGIONID(); RUN;</pre>
GETPROFILEINFO("<user_id>","<field_name>")	For the user identified by <user_id> returns the value of <field_name>	<p>Example:</p> <pre>PROC PRINT; PUT GETPROFILEINFO(GETUSERID(),"USERLIB PROFILE"); RUN;</pre>
GETPRIVATEEGIONID()	Returns the ID of the PRIVATE region.	<pre>PROC PRINT; PUT GETPRIVATEREGIONID(); RUN;</pre>
GETPUBLICREGIONID()	Returns the ID of the PUBLIC region.	<pre>PROC PRINT; PUT GETPUBLICREGIONID(); RUN;</pre>
GETSHAREDREGIONID()	Returns the ID of the SHARED region	<pre>PROC PRINT; PUT GETSHAREDREGIONID(); RUN;</pre>

GETUSERID()	Returns the unique identification number of the current user	Refer to examples:  <a href="#">GETUSERID()_GETUSERINFO()</a>  <a href="#">PROC EFRONTMENUS - GETUSERPROFILE - GETUSERID</a>
GETUSERINFO("<user_id>","<field_name>")	For the user identified by <user_id> returns the value of <field_name>   Only applies to eFront Cube programs	Example:  PROC PRINT;  PUT "My current profile label is: " & GETUSERINFO(GETUSERID(),"USERLIBPROFILE");  RUN;  Refer to example (list of meaningful <field_name> values):  <a href="#">GETUSERID()_GETUSERINFO()</a>
GETUSERPROFILESLIST ("<user_id>")	Returns the list of unique identifiers of all the user's profile (separated by ";")	Example:  %LET MyProfiles = GetUserProfilesList(GetUserID());  TRACE %MyProfiles;
GETUSERPROFILE ("<user_id>")	Returns the unique identification number of the user's current profile   Only applies to eFront Cube programs	Examples:  %LET MyProfileName = GETUSERPROFILE(GETUSERID())  <a href="#">PROC EFRONTMENUS - GETUSERPROFILE - GETUSERID</a>
GUID()	Returns a unique identification number (Global Unique Identifier), used in the eFront database	INVESTOR_ID = GUID();
LAG(<column>)	Selects the value of the preceding line.   Performs the same as PARENT(<column>)	

LOCALIZE(<column>)	<p>Returns the column label in the language that corresponds to the user's local settings.</p>	<p>Example:</p> <pre>... COLUMN IRR LABEL="IRR(inv){F}TRI(inv)";  COLUMN IRR_LABEL;  IRR_LABEL = LOCALIZE(IRR);  ...</pre> <p>Result:</p> <p>If the user's local language = "English", the value of IRR_LABEL = "IRR(inv)"</p>
LOOKUP("<table_name>" , <code>)	<p>Function that applies specifically to the list of options being available for additional fields.</p> <p>Therefore you can only use this function if additional fields have been defined, and if a list of options has been defined for the additional field you refer to.</p> <p>A list of options is a list of &lt;code&gt; &lt;label&gt; entries. The &lt;code&gt; refers to the value sent to the system if the user selects an option. The &lt;label&gt; refers to the label that displays in the drop-down list from which the user selects an option.</p> <p>The LOOKUP() function returns the value of the &lt;label&gt; column that corresponds to the value of the &lt;code&gt; column being referenced to in the table &lt;table_name&gt;</p>	<p>Refer to example:</p> <p><a href="#">LOOKUP()_LOOKUPCODE()</a></p>

<code>LOOKUPCODE("&lt;table_name&gt;" , "&lt;label&gt;")</code>	<p>Function that applies specifically to the list of options being available for additional fields.</p> <p>Therefore you can only use this function if additional fields have been defined, and if a list of options has been defined for the additional field you refer to.</p> <p>A list of options is a list of <code>&lt;code&gt; &lt;label&gt;</code> entries. The <code>&lt;code&gt;</code> refers to the value sent to the system if the user selects an option. The <code>&lt;label&gt;</code> refers to the label that displays in the drop-down list from which the user selects an option.</p> <p>The <code>LOOKUPCODE()</code> function returns the value of the <code>&lt;code&gt;</code> column that corresponds to the value of the <code>&lt;label&gt;</code> column being referenced to in the table <code>&lt;table_name&gt;</code></p>	<p>Refer to example:</p> <p><a href="#">LOOKUP()_LOOKUPCODE()</a></p>
<code>LOOKUPDYN("&lt;table_name&gt;","&lt;column_name&gt;","&lt;code&gt;")</code>	<p>Retrieves the label for user lookup defined in a dynamic table.</p> <p>Options:</p> <p><code>&lt;table_name&gt;</code> refers to the name of the dynamic table.</p> <p><code>&lt;column_name&gt;</code> refers to the name of the column that contains the lookup.</p> <p><code>&lt;code&gt;</code> refers to the code of the lookup.</p> <p> Not available for eFront Cube programs</p>	<pre>%LET Label = LOOKUPDYN("Table", "METATEXT1", "1");</pre>

NZ(<value-1>, <value-2>)	Returns <value-1> if <value-1> is not NULL, else, returns <value-2>	Refer to example: <a href="#">DATA - SET - IN</a>
PARENT(<column>)	Selects the value of the preceding line.   Performs the same as LAG(<column>)	
PROFILE("<packageoption>","<defaultvalue>")	Returns the package option value for the current profile, or the default value.	Example:  PROC PRINT;  PUT PROFILE("PCKAJXPORTAL.SKIN","METRO") ;  RUN;  Result:  The skin value is returned (metro, aerial or legacy) or METRO, if the first parameter isn't defined.
REGISTRY(<type>, <key>, <name>)	Not authored yet.   <type> must be "ACCOUNT", "USER", "TEMP" or "MEM"	Examples:  REGISTRY("ACCOUNT","AUDITING","TABLES");  Result: the list of tables being audited (e.g. ADMREGISTRY;ADMUSER;ajxrepregistry;VC FUND)  REGISTRY("ACCOUNT","AUDITING","ENABLED_COLS");  Returns: 1  REGISTRY("USER","AJXPORTAL","CONTENT");  Returns: the application language

SETLANGUAGE(<language>)	Forces the language for localization of the strings in the report (like caption of reference tables in views). The user language is used for localization by default.   Not available with eFront Cube programs.	%LET LANGUAGE = SETLANGUAGE("J"); PROC PRINT;  PUT LOCALIZE("Hello World{J}おはようございます");  PUT %LANGUAGE;  RUN;  %LET LANGUAGE = SETLANGUAGE(%LANGUAGE);  PROC PRINT;  PUT LOCALIZE("Hello World{J}おはようございます");  PUT %LANGUAGE;  RUN;  Will return the following text:  おはようございます A Hello World J
-------------------------	---	--

<pre>SETPDFMETADATA(   &lt;file_name&gt;,"&lt;table_name&gt;","&lt;key_column&gt;" ,   "&lt;value_column&gt;")</pre>	<p>Adds custom metadata to an existing PDF file. The metadata is sourced from a eFront Script table, using set of keys and a set of values.</p> <p>Note that the PDF file must be located on the application server. The program will not compile if the PDF file is attached to the program or located elsewhere.</p> <p> Not available for eFront Cube programs</p>	<pre>DATA WORK.TMP; SQL "SELECT A.FUND, A.ABBREVIATION FROM VCFUND A WHERE ??FILTER;"; COLUMN FUND LABEL="Fund"; COLUMN ABBREVIATION LABEL="Abbr"; RUN;  %LET MERGED_FILE=SETPDFMETADATA("D:\eFront\download\Test.pdf", 'WORK.TMP', "Fund", "Abbreviation");  PROC PRINT; PUT %MERGED_FILE; RUN;  Result:  The custom properties of <b>Test.pdf</b> are updated with data from the FUND and ABBREVIATION columns of the VCFUND table. Note that, because ??FILTER is used, the program uses only the data that the user has access rights to.</pre>
<pre>TABLEINFO("table",   "&lt;info&gt;")</pre>	<p>Returns information about specified table. Information can be : "ID", "USER", "DATE", "CLASS", "PATH", "NAME"</p>	<p>Example:</p> <pre>%LET CREATION_DATE = TABLEINFO   ("FV_T_INFO_FUND", "DATE");</pre>
<pre>TABLENCOL("table")</pre>	<p>Returns the number of columns of a data table.</p>	
<pre>TABLENOBS(table)</pre>	<p>Returns the number of rows of a data table.</p> <p>Interesting to use in control structures where you need to test if a table has some content.</p>	<p>Refer to example :</p> <pre>PROC PRINTFORM - TABLENOBS() - %IF %THEN %END</pre>

## 4.9.6. Operators

Operators with higher precedence are evaluated before operators with relatively lower precedence. Operators in the same cell have equal precedence. When operators of equal precedence appear in the same expression, a rule must govern which is evaluated

first. All binary operators except for the assignment operators are evaluated from left to right; assignment operators are evaluated right to left.

**Macro operators** are prefixed by %.

Precedence	Operator type	Operators	Example
1	unary	NOT -	-12
2	exponential, multiplicative, modulo	* / ^ ** MOD	%FOR - %NEXT - MOD
3	additive	+ -	7-3
4	comparative	IS NULL IS NOT NULL LIKE NOT LIKE IN, %IN	%LET - %FOR - %NEXT  IN works only with collections (you get them by using PARM with XID or XPICKID), not with strings

		FASTIN	<p>Like IN, FASTIN compares the value of the column with the values in the collection. However, the FASTIN operator applies stronger type checking to speed up comparison process. The difference between the two operators is in the following:</p> <ul style="list-style-type: none"> <li>• Empty string and null are considered equal with IN and different with FASTIN;</li> <li>• NaN double (Not a Number) and null are considered equal with IN and different with FASTIN;</li> <li>• Value of type <i>float/double</i> and value of type <i>integer</i> : 1000 as double is considered equal to 1000 as integer with IN but different with FASTIN);</li> <li>• Boolean True and Integer 1 are considered equal with IN and different with FASTIN;</li> <li>• Boolean False and Integer 0 considered equal with IN and different with FASTIN.</li> </ul>
			 <b>CAUTION</b> FASTIN is available only in eFront Cube.
		NOT ... IN	<a href="#">PROC SORT - (WHERE - NOT...IN)</a>
		BETWEEN(,)	
		NOT BETWEEN(,)	
5	concatenation	+ - &	<a href="#">%FOR - %NEXT - MOD</a>
6	relational	<> <= >= < = >	
7	logical	AND	
		OR	

#### 4.9.6.1. Comment

- LIKE can be used together with the placeholder "%" in case you build a character string with which to compare column values.
- Here is an example for using the operator BETWEEN:

```
SET TEMP1.[EFFV_CDW_T_COMPANY_TRANSACTION](WHERE CASH
BETWEEN(1,1000000));
```

#### 4.9.7. Special values

Value	Description	Example
TRUE		
FALSE		
NULL   EMPTY   MISSING   NOTHING   ".."	used together with IS or IS NOT	IF name IS NOT NULL THEN ...

#### 4.9.8. Specifying values or value ranges

Each occurrence of value-or-range is either one of the following:

<value_or_range>	Possible values
<value>	A single value, such as 15 or "instrument"
	Enclose character values in single or double quotation marks.
	Use the keyword OTHER as a single value. OTHER matches all values that do not match any other value or range.
	Value specifications are case-sensitive
<range>	A list of values, for example, 14-98 or 'A'-'Z'
	Enclose character strings in quotation marks
	You can use LOW or HIGH as one value in a range, and you can use the range LOW-HIGH to encompass all values.
	Example :
	LOW - 'Z' or 35 - HIGH
	LOW-HIGH ='#,##0; (#,##0); 0';
	Use the less than (<) symbol to exclude values from ranges.
	If you are excluding the first value in a range, then put the < after the value. If you are excluding the last value in a range, then put the < before the value. For example, the following range does not include 0:
	0<-100
	Likewise, the following range does not include 100:
	0-<100

	<p>If a value at the high end of one range also appears at the low end of another range, and you do not use the &lt; noninclusion notation, then PROC FORMAT assigns the value to the first range. For example, in the following ranges, the value AJ is part of the first range:</p> <p>'AA'-'AJ'=1 'AJ'-'AZ'=2</p> <p>In this example, to include the value AJ in the second range, use the noninclusive notation on the first range:</p> <p>'AA'-&lt;'AJ'=1 'AJ'-'AZ'=2</p>
	<p>If you overlap values in ranges, then PROC FORMAT returns an error message unless, for the VALUE statement, the MULTILABEL option is specified. For example, the following ranges will cause an error:</p> <p>'AA'-'AK'=1 'AJ'-'AZ=2</p>

## 4.10. Style attributes and values

When working with styles you need to identify:

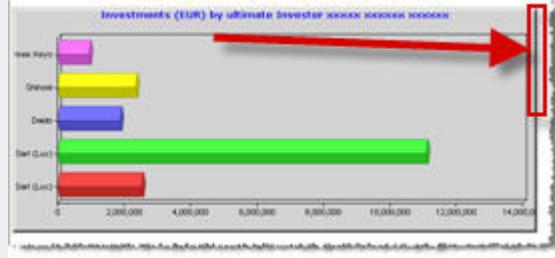
- [Items to be styled](#)
- [Chart style attributes and values](#)
- [Text style attributes and values](#)

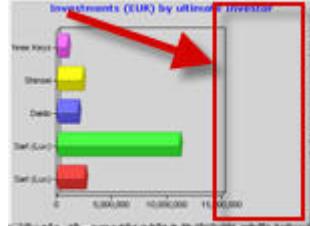
### 4.10.1. Items to be styled

Step	Items to be styled	
PRINT	BANNERFILE	N
	BANNER	OBS
	COLHEADER	TITLE
	DATA	TITLE1
	GRANDTOTAL	TOTAL
	HEADER	
PRINTCOL	DATA TITLE TITLE1	
TABULATE	BANNERFILE BANNER BOX CLASS	DATA HEADER TITLE TITLE1
GCHART	DONUT DONUT3D HBAR HBAR3D PIE	PIE3D TITLE TITLE1 VBAR VBAR3D

## 4.10.2. Chart style attributes and values

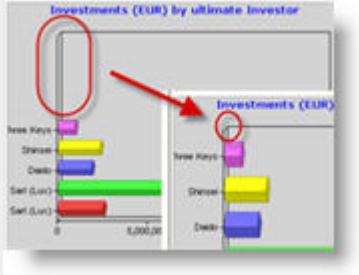
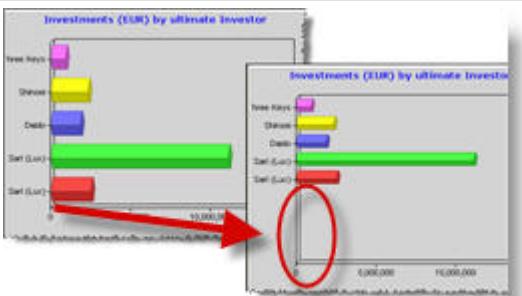
This is only a limited list of chart style attributes and values, mainly attributes that apply to bar charts and plot & line chart. To get the complete list of possible attributes, please use the **eFront Report** editor. When units are required, we recommend you use **pixel units** instead of percentages.

Style options	Values	Description
BARCOMBINEMETHOD= <value>	<value>: SIDE   STACK   PERCENTAGE	In a bar chart, specifies the way to combine bars
CHARTCOLOR= <color>	<color>: quoted string that defines the color or Hex codes included in HTML applicable color charts such as the Browser Safe Colors	Color of chart
CHARTEDGE=<value>	<value>: integer	Width of the shaded edge 

CHARTEXTCOLOR=<color>	<color>: quoted string that defines the color or Hex codes included in HTML applicable color charts such as the Browser Safe Colors	Color of chart edge
CHARTROUNDEDFRAME=<value>	<value>: TRUE   FALSE	Chart with/without rounded frame
HEIGHT=<value>	<value>: integer	height of chart
LABELFORMAT=" <text> {VALUE} &lt;n&gt;,&lt;options&gt;} &lt;text&gt;"</text>	<text>: free text  <n>: 1   2  1: specifies the Y-axis value  2: specifies the X-axis value  <options>: digit separator: ','   '.'	In a bar chart, defines the text to display above each bar
LEGEND=True LEGENDVERTPOS=<value>	<value>: integer  Allows to enlarge the right margin of the chart	

LINECOLOR=<value>	<color>: quoted string that defines the color or Hex codes included in HTML applicable color charts such as the Browser Safe Colors	In a plot & lines chart, color of line
LINEWIDTH=<value>	<value>: integer	In a plot & lines chart, width of line
LINESTYLE=<value>	FULLLINE   DASHLINE   DOTLINE   ALTDASHLINE   3D	In a plot & lines chart, type of line
LINELABELFORMAT= "<text> {VALUE <n>,<options>} <text>"	<text>: free text  <n>: 1   2  1: specifies the Y-axis value  2: specifies the X-axis value  <options>: digit separator: ','   '.'	Adds text to the plots used in a plot & lines chart  Example:  {VALUE 2,.} USD applied to "123456.789", displays on the screen "123,456.78 USD"
SERIEMULTICOLORS=<value>	<value>: TRUE   FALSE	In a bar chart, specifies that bars are of different color

SERIESSHAPE=<value>	<value>: DIAMOND   SQUARE   PLUS   CIRCLE   ETOILE   TRIANGLE   RIGHT TRIANGLE   LEFT TRIANGLE   INVERTED TRIANGLE   POLYGON   POLYGON2	In a bar chart, specifies the shape of bars
TITLE1POSITION = <value>	<value>: integer  1: Bottom-left, 2: Bottom-center, 3: Bottom-right, 4: Middle-left, 5: Middle-center, 6: Middle-right, 7: Top-right, 8: Top-center, 9: Top-left	Position of title 1
WIDTH = <value>	<value>: integer	Width of chart, useful to create enough room for labels
XAXISWIDTH =<value>	<value>: integer  Specifies the width of the X axis	
XAXISANGLE=<value>	<value>: integer	Angle of labels on x-axis

XAXISTOPMARGIN=<value>	<value>: integer  Specifies the space between the top bar of the X-Axis, and the top line of the chart	
XAXISBOTTOMMARGIN=<value>	<value>: integer  Specifies the space between the top bar of the X-Axis, and the bottom line of the chart	
XAXISFONTSIZE=<value>	<value>: integer  Specifies the font size of X-Axis labels	
XAXISFORMAT="150%"		
YAXISTITLE=<value>	<value>: string  Specifies the title of the Y-axis	
YAXISANGLE=<value>	<value>: integer	Angle of labels on y-axis

#### 4.10.3. Text style attributes and values

Style attribute	Values	Description
ALIGN = <value>	RIGHT   LEFT   CENTER	Alignment of cell content

VALIGN = <value>	RIGHT   LEFT   CENTER	
PADDING = <value>	<number> <unit>, where <unit> = cm   mm   in   pt   pc   px	Space between cell border and cell content
BORDER	NONE   <number> <unit>, where <unit> = cm   mm   in   pt   pc   px	Thickness of border line
BACKGROUND = <value>	YELLOW   ORANGE   PINK   RED PURPLE   VIOLET   CYAN   BLUE   GREEN   BROWN   GRAY   BLACK  ...  or Hex codes included in HTML applicable color charts such as the Browser Safe Colors	Color to be used as background
FOREGROUND = <value>	YELLOW   ORANGE   PINK   RED PURPLE   VIOLET   CYAN   BLUE   GREEN   BROWN   GRAY   BLACK  ...  or Hex codes included in HTML applicable color charts such as the Browser Safe Colors	Color to be used as foreground
FONT_FACE = <value>	<font_name>	Font to be used to format text
FONT_SIZE = <value>	<number> <unit>, where <unit> = cm   mm   in   pt   pc   px	Size of font
FONT_STYLE = <value>	ITALIC   ROMAN	Style of font
FONT_WEIGHT = <value>	BOLD   MEDIUM   LIGHT	Weight of font

- If you apply more than one style attribute, use blank spaces as separator.

## 4.11. Flags

List of available flags:

- `_OUTPUT_flag`
- `IN flag`

### 4.11.1. `_OUTPUT_flag`

#### 4.11.1.1. Description

Valid in: DATA step. Possible values : TRUE | FALSE.

#### 4.11.1.2. Syntax

```
DATAoutput_table (</ table_options>);
```

```
IF <condition_1> THEN
```

```
    _OUTPUT_ = TRUE;
```

```
ELSE
```

```
    _OUTPUT_ = FALSE;
```

```
END;
```

```
RUN;
```

#### 4.11.1.3. Comment

- If `_OUTPUT_ = TRUE`: eFront Script writes the data line to the output\_table.
- If `_OUTPUT_ = FALSE`: eFront Script writes no data to the output\_table.
- If you use a DATA step together with an OUTPUT statement, `_OUTPUT_` is set to FALSE, as soon as a DATA step reads the OUTPUT statement.
- You can avoid writing a condition, if you use the `_OUTPUT_` flag together with the `INflag`. As both flags are of boolean type, you can instantiate the `_OUTPUT_` flag using the `_OUTPUT_` flag. (Example)
- You can use `_OUTPUT_= NOT.`

#### 4.11.1.4. Error messages

Message	Explanation
---------	-------------

### 4.11.2. IN flag

#### 4.11.2.1. Description

Valid as **table option** in the MERGE and SET statements. Possible values: TRUE | FALSE. Allows you to flag the table, a row comes from. If used, it must be the First element on qualifying a table and then may be tested (If..THEN...ELSE controls.) so you can check where a row comes from and trigger corresponding processing.

#### 4.11.2.2. Syntax

DATAoutput\_table (</ table\_options>);

SET | MERGEinput\_table\_1 (IN=flag\_1) input\_table\_2 (IN=flag\_2);

BYcol;

Ifflag\_1 <operator> flag\_2 THEN

</ subordinate statements>;

OUTPUT;

END;

RUN;

#### 4.11.2.3. Comment

- IN is TRUE if the data line read is part of the MERGE, therefore flag\_1 or flag\_2 are set to TRUE. This allows you later on, in the conditional loop to build a test on flag\_1 and flag\_2.

#### 4.11.2.4. Error messages

Message	Explanation
ERROR(0113): Variable name expected	It could be that the value assigned to the IN flag is wrong

## 4.12. Variables

List of available **eFront Script** variables:

- Macro variables
- Magic variables
- The variable: `ALL`
- The variable: `_LEVEL_`
- The variable: `_URL_`
- The variable: `%DEBUG`

### 4.12.1. Macro variables

#### 4.12.1.1. Description

Macro variables are global in scope, and can be used from within DATA or PROC steps, **macro statements** or **macro controls**, that follow the macro variable definition. Macro variables can also be used from within XML templates being used to generate **XML reports**.

Macro variables are defined by the macro statements:

- `%LET` statement
- `%PARAM` statement

There are two types of macro variables:

- Macro variables that point to a **value** or a **list of values**: prefix the macro variable name by `%` when using it in a DATA or PROC steps, **macro statements** or **macro controls**.
- Macro variables that point to a **table name** or a **column name** prefix the macro variable name by `@` when using it in a DATA or PROC steps, **macro statements** or **macro controls**.

Use the macro variable `%DEBUG` to debug your programs.

## 4.12.2. Magic variables

### 4.12.2.1. Description

Valid in: **LIBNAME** and **%INCLUDE** statements. Point to reserved **eFront Report** regions. A collection of magic variables are available:

\{SYSTEM}

\{SHARED}

\{PRIVATE}

\{PUBLIC}

\{ROOT}

\{INFO}

- **SYSTEM**: refers to a region, also called region 0, that is visible to all accounts installed on the same server. Only users with the profile **Remote Admin** can modify data of this region (**Write** access). All other users have **Read only** access.
- **SHARED**: refers to the shared region, a folder that contains standard programs and standard tables shared by all users.
- **PRIVATE**: refers to the private region, a folder that contains data owned by the current user.
- **PUBLIC**: refers to the public region, a folder that contains data shared by all users.
- **INFO**: refers to the Information region.
- **ROOT**: refers to the region where the eFront Script program is located.

When **eFront Script** encounters a magic variable, it automatically instantiates it with the path being used to access the reserved region. Paths can be defined as a system option in the **eFront Report Administration** section.

### 4.12.2.2. Syntax

%INCLUDE "\{SHARED\}\program\_name";

LIBNAME**library\_alias** "\{PUBLIC\}\library\_name";

### 4.12.2.3. Error messages

Message	Explanation
---------	-------------

## 4.12.3. The variable: ALL

### 4.12.3.1. Description

Valid in: TABLE statements. Summarizes information from a class of values.

### 4.12.3.2. Syntax

```
PROC TABULATE DATA = table_to_process (</ table_options>;
```

```
CLASS col_1 col_n;
```

```
VAR var_1 var_n;
```

```
TABLE col_to_summarizeALL = "label",
```

```
col_to_summarizeALL = "label"*
```

```
</ functions>;
```

```
RUN;
```

### 4.12.3.3. Comment

- When you summarize values, eFront Script adds a new column to the table that contains the result. If you don't specify a column header, the default header = ALL. To modify the default column header, assign a label to ALL.

### 4.12.3.4. Error messages

Message	Explanation
---------	-------------

## 4.12.4. The variable: \_LEVEL\_

### 4.12.4.1. Description

Valid, if used together with the RECURSE statement. When calculating dependancy links, the level of hierarchy is automatically assigned to the \_LEVEL\_ variable.

### 4.12.4.2. Syntax

DATAoutput\_table (</ table\_options>);

RECURSEinput\_table\_1 (</ options>) input\_table\_2 (</ options>);

COLUMNcol\_name (</ options>);

LEVEL = \_LEVEL\_;

...;

RUN;

### 4.12.4.3. Comment

- You can for example reference the LEVEL variable in the PROC PRINT step, and print only data of the level being specified.

### 4.12.4.4. Error messages

Message	Explanation
---------	-------------

## 4.12.5. The variable: \_URL\_

### 4.12.5.1. Description

Variable used to define a link to a destination entity, either to a data table or to a cell in the data table. Use this variable together with the [EVENT step](#).

### 4.12.5.2. Syntax 1

//Defines a link to a table

```
_URL_="TXT_REPORT.FOLDER=Folder,Var";
```

#### 4.12.5.3. Syntax 2

//Defines a single link anchored in a particular cell within a table

```
_URL_="TXT_REPORT.FOLDER=Folder,Var,Column";
```

#### 4.12.5.4. Syntax 3

//Defines several links anchored in a particular cell within a table

```
_URL_="TXT_REPORT.FOLDER=Folder,Var,Column;TXT_REPORT.FOLDER=Folder,Var,Column";
```

#### 4.12.5.5. Options

Option	Description
Folder	Link destination
Var	IQID of destination entity
Column	Cell of destination table

#### 4.12.5.6. Comment

- You can concatenate define more than one destination links for a table cell when defining the value of the \_URL\_ variable! This allows you for example to define for each cell of a table different links, one which points to the investor, another one which points to the company, etc.

### 4.12.6. The variable: %DEBUG

#### 4.12.6.1. Description

%DEBUG is a [macro variable](#). It is a **boolean** variable, whose value is used to **activate or deactivate** [TRACE statements](#) used to debug programs.

#### 4.12.6.2. Syntax

```
%IF %DEBUG %THEN
```

```
TRACE...;
```

%END;

#### 4.12.6.3. Comment

To efficiently use the %DEBUG variable:

1. Define %DEBUG at **System options** level, and set its value to FALSE.
2. Define %DEBUG at **My options** level, and set its value to TRUE if you want to activate Trace statements.

## 4.13. XML reports

- [About XML reports](#)
- [eFront Invest XML Schema Definition](#)

### 4.13.1. About XML reports

It is possible to produce XML reports using the [PROC OFFICE step](#). For the moment, the **eFront Report document editor** doesn't manage the XML templates. Therefore you can only generate XML reports using eFront Script programs. To make generate a report, you need to build a well-formed and valid XML template used as report generation support.

**Well-formed** means, that:

- The XML template used has exactly one root element, also known as document element, i.e. the template structure is enclosed between a root start-tag and a corresponding end-tag, such as this:  
`<Template> ... </Template>`.
- Comments are inserted using the standard syntax that is:  
`<!-- This is a comment. -->`
- The root element must be preceded by an XML declaration, stating the XML version, character encoding and external dependencies information, such as this:  
`<?xml version = "1.0" encoding="UTF_8"?>`.  
If you omit the XML declaration, it is assumed that the document conforms to XML version 1.0.

**Valid** means that the XML template conforms to :

- semantic rules, being defined in an **XML schema**. The XML Schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. The process of checking to see if an XML document conforms to a schema is called validation. Documents are only considered valid if they satisfy the requirements of the schema with which they have been associated. In eFront Script, a valid XML template conforms to the [eFront XML Schema Definition](#). You don't have to use an explicit linkage to apply the eFront XML

Schema Definition to an XML template being processed by the [PROC OFFICE step](#) in eFront Script. The link is established automatically.

## 4.13.2. eFront XML Schema Definition

### 4.13.2.1. Goal

Generate a XML reports using an XML template that conforms to the standard XML syntax rules (well-formedness), and to the eFront XML Schema Definition.

### 4.13.2.2. Specific eFront nodes that can be used

In the following you find the specific nodes to be used to generate a [valid XML file](#). For each node, you find a detailed description (click the node). All the nodes that are being interpreted by the eFront parser start with the prefix **EF\_**:

- EF\_LOOP
- EF\_VALUE
- EF\_G
- EF\_MACRO
- EF\_ATTRIB
- EF\_SELECT
- EF\_CASE
- EF OTHERWISE

### 4.13.2.3. Comments

- Nodes can be nested.

### 4.13.2.4. EF\_ATTRIB

#### Goal

EF\_ATTRIB can be used in an XML template and allows you to create node attributes in the final XML file. The attribute being created is linked to the parent node EF\_ATTRIB is part of. To associate a value with the node attribute being created, use [EF\\_VALUE](#).

#### Syntax

<Node>



```
<EF_ATTRIB NAME = "attrib_name">
</EF_ATTRIB>
</Node>
```

## Node Attributes

Attribute	Description
NAME	<p>required;</p> <p>attrib_name :</p> <p>can be a character string or an expression; if it is an expression, it must evaluate to a single value</p>

### 4.13.2.5. EF\_G

#### Goal

EF\_G can be used in an XML template and allows you to refer to a single value of a specified table. It returns the value of the column for the first line where all conditions apply.

#### Syntax

```
<Node>
```

...

```
<EF_G COLUMN = "colum_name" TABLE =
"table_name" WHERE="condition1;...;conditionN" FORMAT ="format"/>
```

```
</Node>
```

## Node Attributes

Attribute	Description
COLUMN	<p>required;</p> <p>column_name : character string;</p>

TABLE	required; table_name : character string;
WHERE	optional; conditionN := Expression = Expression
FORMAT	optional; format : "dd-mm-yyyy", "#,###.00", ...

## Expression

Condition/Expression	Value	Description
Expression :=	ColumnName	Refers to a column of the table of the current node
Expression :=	?ColumnName	Refers to a column of the table of previous EF_Loop
Expression :=	%MacroName	
Expression :=	True	
Expression :=	False	
Expression :=	'StringValue'	
Expression :=	NumericValue	

### 4.13.2.6. EF\_LOOP - EF\_VALUE

#### Goal

EF\_LOOP and EF\_VALUE can be used in an XML template from with you can generate a valid XML file. EF\_LOOP and EF\_VALUE work together. They contribute to build an XML object for each table entry that conforms to the conditions being expressed. For example, if you take the table of funds as input table, the node EF\_LOOP ensures that as many <FUND> nodes as the table has fund entries (if no restrictive condition is being applied) are created. The node EF\_VALUE allows you associate a particular table column value to a node or a node attribute. EF\_LOOPS can be nested.

#### Syntax

```
<EF_LOOP TABLE = "table_name" WHERE="condition1;...;conditionN">
```

...

<Node>

```
<EF_VALUE COLUMN = "column_name"FORMAT ="format"/>
```

...

```
</Node>
```

...

```
</EF_LOOP>
```

## Node Attributes

Node	Attribute	Description
EF_LOOP	TABLE	required;
EF_LOOP	WHERE	optional; conditionN := Expression = Expression
EF_VALUE	COLUMN	required;
EF_VALUE	FORMAT	optional; format: "dd-mm-yyyy", "#,###.00", ...

## Expression

Condition/Expression	Value	Description
Expression :=	ColumnName	Refers to a column of the table of the current node
Expression :=	?ColumnName	Refers to a column of the table of previous EF_Loop
Expression :=	%MacroName	
Expression :=	True	
Expression :=	False	
Expression :=	'StringValue'	
Expression :=	NumericValue	

### 4.13.2.7. EF\_MACRO

#### Goal

EF\_MACRO can be used in an XML template and allows you to refer to the value of a [macro variable](#).

## Syntax

```
<Node>

...
<EF_MACRO NAME = "macro_name"FORMAT ="format"/>
...
</Node>
```

## Node Attributes

Attribute	Description
NAME	required; macro_name : character string;
FORMAT	optional; format: "dd-mm-yyyy", "#,###.00", ...

## 4.13.2.8. EF\_SELECT - EF\_CASE - EF OTHERWISE

### Goal

EF\_SELECT, EF\_CASE and EF OTHERWISE can be used as control structure in an XML template from which you can generate a valid XML file. The control structure is being contained in the EF\_SELECT node. According to the value taken by the expression that is evaluated in the EF\_SELECT node, you can distinguish different cases (EF\_CASE nodes) to be processed differently. The EF OTHERWISE node is used to cover all the remaining cases that are not particularly specified via a EF\_CASE node. EF\_SELECT nodes can be nested.

## Syntax

```
<Node>

<EF_SELECT EXPR = "expression_to_compare">
<EF_CASE VALUE = "expression1;expressionN">
...

```

```

</EF_CASE>

<EF_CASE VALUE = "expression1;expressionN">
...
</EF_CASE>

EF_OTHERWISE
...
</EF_OTHERWISE>

</EF_SELECT>

</Node>

```

## Node Attributes

Node	Attribute	Description
EF_SELECT	EXPR	required;
EF_CASE	VALUE	required;  expressionN is compared to expression_to_compare according to the following scheme:  expression_to_compare = expression1 OR expression_to_compare = expressionN

## Expression

Condition/Expression	Value	Description
Expression :=	ColumnName	Refers to a column of the table of the current node
Expression :=	?ColumnName	Refers to a column of the table of previous EF_Loop
Expression :=	%MacroName	
Expression :=	True	
Expression :=	False	
Expression :=	'StringValue'	

Expression :=	NumericValue	
---------------	--------------	--

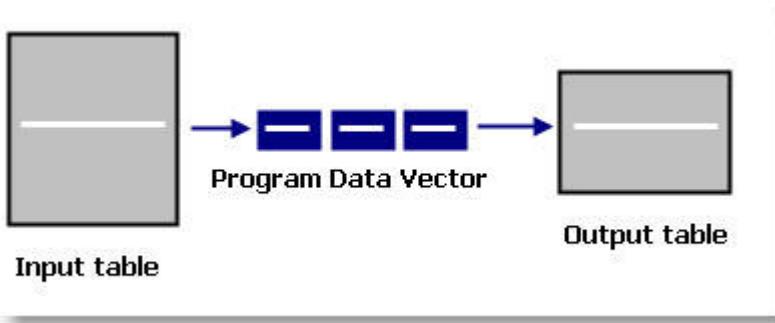
## 4.14. More about language elements

To get a deeper understanding of some of the **eFront Script** processes and concepts, browse through the above list of topics:

- [Understanding the DATA step](#)
- [Understanding the MERGE statement](#)
- [Understanding the SET statement](#)
- [Understanding the TABLE statement](#)
- [Understanding global statements](#)
- [Understanding macro statements](#)
- [Using styles](#)
- [Using formats](#)

### 4.14.1. Understanding the DATA step

The [Data step](#) operates as follows:

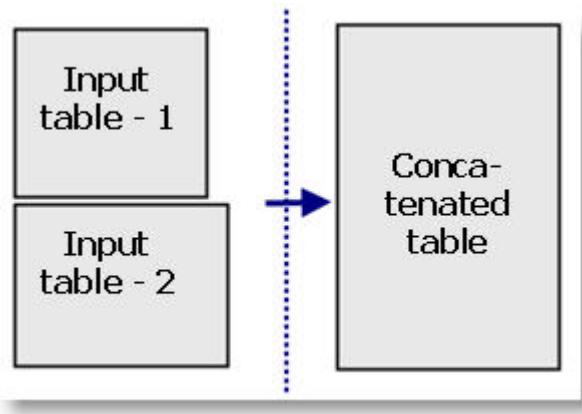


**eFront Script** processes the input table record by record. **eFront Script** reads each record into the **Program Data Vector** (PDV), where data filters and statements are applied, and writes the processing results into the output table.

You only need to program the record reading loop when using the **DATAstep** without **SET**, **MERGE** or **INFILE** statements. These statements automatically operate in loop-mode. To explicitly transfer data records from the **PDV** to the Output table, use the [OUTPUT](#) statement.

#### 4.14.2. Understanding the SET statement

When using the SET statement with more than one input\_table, **eFront Script** concatenates tables, i.e. one is appended to the other. Therefore tables need to have a similar structure.



The number of columns equals the sum of all the columns of all the tables, with the same columns being regrouped into one. If values are missing, cells are left without values.

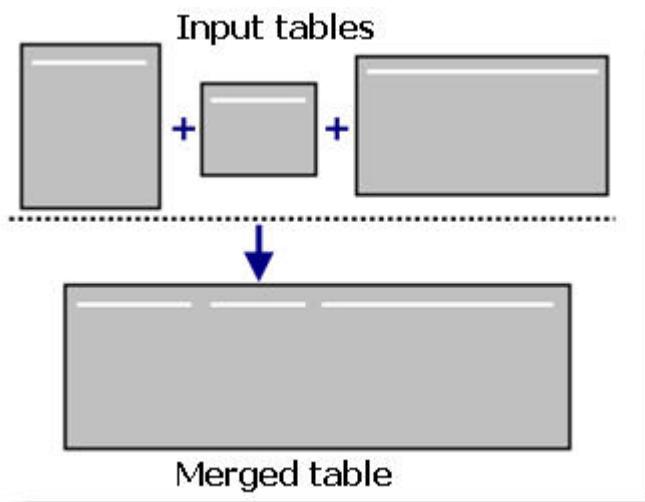
#### 4.14.3. Understanding the MERGE statement

When merging two or more tables, 2 basic merge types can be applied:

- One-to-one merging
- Match merging

One-to-one merging

The one-to-one merge operates as follows :



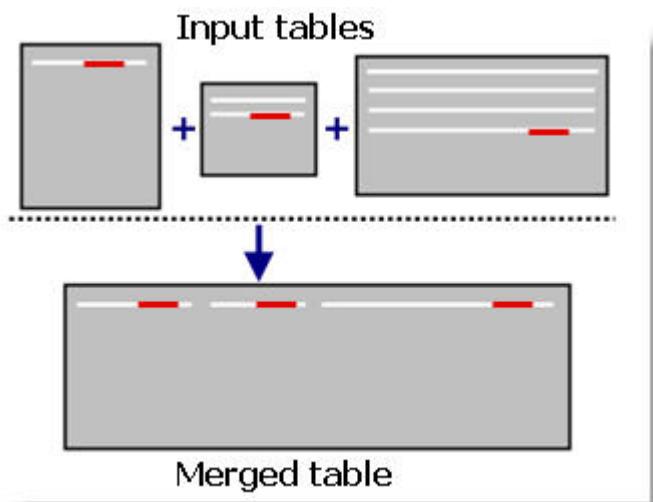
Two or more input tables are merged into one single table where each line = the concatenation of the lines of the input tables. To perform a one-to-one merge, use the MERGE statement **without** a BY statement.

The number of columns equals the sum of all the columns of all the tables, with the same columns being regrouped into one.

The number of lines of the merged table equals the number of lines of the longest table. Identical values are not eliminated.

#### Match merging

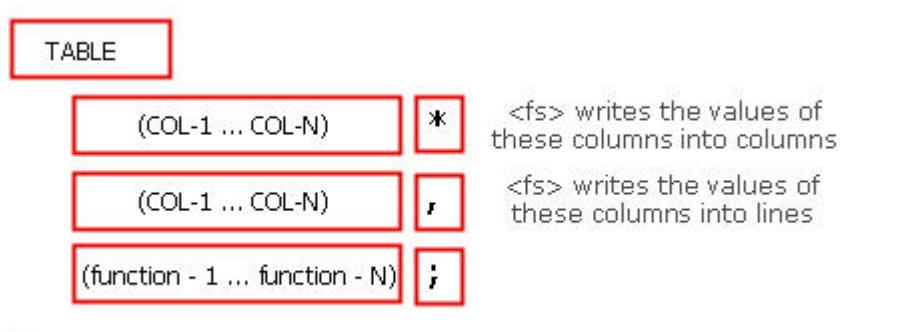
Match-merging operates as follows :



Two or more input tables are merged into one single table where each line = the concatenation of the lines of the input tables. Lines are chosen according to the **values of a common variable**. To perform a match-merge, use a **BY statement immediately after the MERGE statement**. The variables in the BY statement must be common to all data sets. You can only use one BY statement in a MERGE statement.

#### 4.14.4. Understanding the TABLE statement

The TABLE statement defines how column and lines are organized in the statistics table. The statement structure could be schematized in the following way:



Here is an example :

```
//Create the alias that points the existing table location
LIBNAME TEST "\\\EFront_SUP.2 (Privé)\TEST\EXAMPLES\TablesFormation";

//Define the statistical data table

PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;
CLASS USER_ID;
VAR UNIT_PRICE;

TABLE USER_ID, UNIT_PRICE*(N SUM);

RUN;
```

The TABLE statement calculates for each value of the CLASS USER\_ID statistical information about the UNIT\_PRICE of items being purchased.

According to the separator (comma or star) being chosen after each variable (variable group), the statistics table has different aspects :

Statement	Result																								
TABLE USER_ID, UNIT_PRICE* (N SUM);	<table border="1"> <thead> <tr> <th colspan="2"></th> <th colspan="2">UNIT_PRICE</th> </tr> <tr> <th colspan="2"></th> <th>N</th> <th>SUM</th> </tr> </thead> <tbody> <tr> <td rowspan="4">USER_ID</td> <td>101</td> <td>4</td> <td>19450,3</td> </tr> <tr> <td>102</td> <td>2</td> <td>6441,6</td> </tr> <tr> <td>103</td> <td>1</td> <td>7501</td> </tr> <tr> <td>105</td> <td>1</td> <td>20030,2</td> </tr> </tbody> </table>			UNIT_PRICE				N	SUM	USER_ID	101	4	19450,3	102	2	6441,6	103	1	7501	105	1	20030,2			
		UNIT_PRICE																							
		N	SUM																						
USER_ID	101	4	19450,3																						
	102	2	6441,6																						
	103	1	7501																						
	105	1	20030,2																						
TABLE USER_ID* UNIT_PRICE, (N SUM);	<table border="1"> <thead> <tr> <th colspan="3"></th> <th>N</th> <th>SUM</th> </tr> <tr> <td rowspan="4">USER_ID</td> <td>101</td> <td>UNIT_PRICE</td> <td>4</td> <td>19450,3</td> </tr> <tr> <td>102</td> <td>UNIT_PRICE</td> <td>2</td> <td>6441,6</td> </tr> <tr> <td>103</td> <td>UNIT_PRICE</td> <td>1</td> <td>7501</td> </tr> <tr> <td>105</td> <td>UNIT_PRICE</td> <td>1</td> <td>20030,2</td> </tr> </thead> </table>				N	SUM	USER_ID	101	UNIT_PRICE	4	19450,3	102	UNIT_PRICE	2	6441,6	103	UNIT_PRICE	1	7501	105	UNIT_PRICE	1	20030,2		
			N	SUM																					
USER_ID	101	UNIT_PRICE	4	19450,3																					
	102	UNIT_PRICE	2	6441,6																					
	103	UNIT_PRICE	1	7501																					
	105	UNIT_PRICE	1	20030,2																					
TABLE USER_ID UNIT_PRICE* (N SUM);	<table border="1"> <thead> <tr> <th colspan="4">USER_ID</th> <th colspan="2">UNIT_PRICE</th> </tr> <tr> <th>101</th> <th>102</th> <th>103</th> <th>105</th> <th>N</th> <th>SUM</th> </tr> </thead> <tbody> <tr> <td>N</td> <td>N</td> <td>N</td> <td>N</td> <td>8</td> <td>53423,1</td> </tr> <tr> <td>4</td> <td>2</td> <td>1</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	USER_ID				UNIT_PRICE		101	102	103	105	N	SUM	N	N	N	N	8	53423,1	4	2	1	1		
USER_ID				UNIT_PRICE																					
101	102	103	105	N	SUM																				
N	N	N	N	8	53423,1																				
4	2	1	1																						

TABLE UNIT_PRICE, USER_ID* (N SUM);  TABLE UNIT_PRICE* USER_ID, (N SUM);	<table border="1"> <thead> <tr> <th colspan="2"></th><th colspan="4">USER_ID</th><th colspan="2"></th></tr> <tr> <th colspan="2">101</th><th colspan="2">102</th><th colspan="2">103</th><th colspan="2">105</th></tr> <tr> <th>N</th><th>SUM</th><th>N</th><th>SUM</th><th>N</th><th>SUM</th><th>N</th><th>SUM</th></tr> </thead> <tbody> <tr> <td>UNIT_PRICE</td><td>4</td><td>19450,3</td><td>2</td><td>6441,6</td><td>1</td><td>7501</td><td>1</td><td>20030,2</td></tr> </tbody> </table>										USER_ID						101		102		103		105		N	SUM	N	SUM	N	SUM	N	SUM	UNIT_PRICE	4	19450,3	2	6441,6	1	7501	1	20030,2
		USER_ID																																							
101		102		103		105																																			
N	SUM	N	SUM	N	SUM	N	SUM																																		
UNIT_PRICE	4	19450,3	2	6441,6	1	7501	1	20030,2																																	
		101		4		19450,3																																			
UNIT_PRICE	USER_ID	102		2		6441,6																																			
		103		1		7501																																			
		105		1		20030,2																																			

#### 4.14.5. Understanding global statements

Global statements are global in scope and apply to any DATA or PROC steps they precede. The following global statements are available:

- %INCLUDE statement
- LIBNAME statement

#### 4.14.6. Understanding macro statements

Macro statements allow you to customize **eFront Script** programs. Through macro-variables being declared using macro statements, you are able to pass varying values to **eFront Script** programs. Values that you might define through constants (%LET statement), or values that you might request interactively from the user (%PARAM statement). You can also include another program (%INCLUDE statement).

Macro statements start always with the % character. To reference macro-variables from within an **eFront Script** program, prefix the macro variable name with the % character.

When macro statements are being used, **eFront Script** compiles first the macro statements, and then the **eFront Script** program.

The following macro statements are available:

- %INCLUDE statement
- %LET statement
- %PARAM statement

## 4.14.7. Create import mask

1. Access the service **Administration > Guide > Tools > Import > Import**.  
The import manager opens.
2. Create an import mask.
3. Enter the required information.  
Here is an example.

**< Import data**

Region: EFRONT\_ADMIN2(Private)

Description:

**Step 1. Define parameters and click on button 'Load'**

First line of data	2	Check nb columns
Last line of data	1000	<input checked="" type="checkbox"/> Read Header
Separator	Comma	<input type="checkbox"/> Log import result
Max number of error (or -1)	1	<input type="button" value="Load"/>
<input type="button" value="Reload File"/>		

**Regional settings used for import**

Model	(use current)	Digit grouping symbol	Space (10 000)
Decimal symbol	,	Date Format	DD/MM/YYYY
Time Format	24 Hours		

**Step 2. Associate column in import file and column in database**

Imported name	<input type="button" value="&lt;&gt;"/>	Name => fields	Associated Field	Conversion(*1)	Ignore if...(*2)
Given sample	<input type="button" value="&lt;&gt;"/>				

(\*1) Select conversion rules (Code = native storage value, Libelle = displayed value, Table = use conversion table, Expression = convert imported value with an expression).  
(\*2) Enter NULL, - ignore row if field is null, NOT NULL - ignore row if field is not null, or value - ignore row if field is equal to value

**Step 3. when you are ready, press on button 'Import'**

Default region:

Administration

4. Click **Save & Close**.

## 4.14.8. Using styles

You can customize the appearance of your output tables, using styles. You can define styles:

- on a high level through creation of a stylesheet.
- on a low level through definition of particular styles for particular items.

To create a stylesheet, use the **STYLESCHEET** step. Stylesheets can be applied to all the procedures of a program, or to parts of it.

To define and apply a particular style to a table item, use the **STYLE** option.

Low-level style definitions always override high-level style definitions.

## 4.14.9. Using formats

You can customize the appearance of your output tables, using **formats**. You can define formats:

- on a high level through creation of a format sheet.
- on a low level through definition of particular format for particular items.

To create a format sheet, use the [PROC FORMAT](#) step. Formats can be applied to all the columns of a program.

Low-level format definitions always override high-level format definitions.

Any later format specification for a variable, using either the FORMAT statement or the FORMAT option, always overrides the previous one.

## 5. Examples - code snippets

Throughout the syntax reference topics you find code snippets that illustrate syntax elements. Find below the total list of examples, ordered by programming step:

Step being illustrated	Example
DASHBOARD	
DATA	DATA - COLUMN - OUTPUT
	DATA - DO WHILE
	DATA - DO WHILE - CONTINUE
	DATA - DO WHILE - PUT
	DATA - DO WHILE - PUT - HTML tags
	DATA - FOR TO NEXT - OUTPUT
	DATA - IF THEN ELSE
	DATA - INFILE - COLUMN
	DATA - INFILE - INLINE (1)
	DATA - INFILE - INLINE (2)
	DATA - MERGE
	DATA - MERGE - WHERE
	DATA - MERGE - IN - IF THEN ELSE - OUTPUT
	DATA - RECURSE - IF THEN ELSE
	DATA - RECURSE - IF THEN ELSE - _LEVEL_
	DATA - SET - SELECT WHEN THEN
	DATA - SET - WHERE
PROC CONVERTCURR	PROC CONVERTCURR
PROC EFRONTACCRAULS	PROC EFRONTACCRAULS - %PARAM
PROC EFRONTBLOB	PROC EFRONTBLOB - %PARAM
	PROC EFRONTBLOB - DATA - CLASS - PREFIX - ID
PROC EFRONTFOLDER	PROC EFRONTFOLDER - FOLDER - ID - PROPERTIES
	PROC EFRONTFOLDER - Get portfolio positions
PROC EFRONTIMPORT	DATA - PROC EFRONTIMPORT
PROC EFRONTIMPORT_AJX	PROC EFRONTIMPORT_AJX - %OUTPUTLOG
PROC EFRONTMENUS	PROC EFRONTMENUS - GETUSERPROFILE - GETUSERID

PROC EFRONTTABLES	PROC EFRONTTABLES - TABLES - INDEXES - RELATIONS	
PROC EXPORT	PROC EXPORT (1)	
PROC EXPORTCHART	PROC EXPORTCHART - Generate Graph	
PROC EXPORTEXCEL	PROC EXPORTEXCEL	
PROC FALIBRARY	PROC FALIBRARY - CFFPortfolio.PortfolioCashflows	
	PROC FALIBRARY - HFFundAssets.Liquidity	
PROC FAQUERY	PROC FAQUERY - %LET	
	PROC FAQUERY - Contacts - EntireTable	
PROC FORMAT	PROC FORMAT - COLUMN - FORMAT	
	PROC FORMAT - PICTURE - large negative numbers	
	PROC FORMAT - PICTURE	
	PROC FORMAT - VALUE (1)	
	PROC FORMAT - VALUE (2)	
PROC MEANS	PROC MEANS - CLASS - N - LABEL	
	PROC MEANS - SUM	
	PROC MEANS - SUM - MEAN - N	
	PROC MEANS - LEFT() - N	
	PROC MEANS - VAR - MULTIPLE	
PROC OFFICE	PROC OFFICE - Generate XML report	
	PROC OFFICE - Generate muti WORD document generator	
PROC PRINT	PROC PRINT - LABEL	
	PROC PRINT - NOOBS - N	
	PROC PRINT - N - FORMAT - SUM - BY	
	PROC PRINT - SUM - STYLE	
	PROC PRINT - SUM - BY - STYLE	
	PROC PRINT - VAR - .SUM - STYLE	
	PROC PRINT - VAR - .MULTIPLE	
	Example - IRR CASH FLOWS BY INSTRUMENT	
PROC PRINTCOL	PROC PRINTCOL - FLOW - STYLESHEET	
PROC PRINTFORM	PROC PRINTFORM - TITLE - VAR	
PROC SENTTOIC	PROC SENDTOIC	
PROC SETFILTER	ROC SETFILTER - EXTEND - Global Cube and Filtered Session Cube	
PROC SORT	PROC SORT - NODUPKEY - BY	
PROC SQLTABLE	PROC SQLTABLE - (1)	
	PROC SQLTABLE - (2)	

PROC TABULATE	PROC TABULATE - BOX - STYLE	
	PROC TABULATE - TABLE - ALL - N - SUM	
	PROC TABULATE - TABLE - STYLE (1)	
	PROC TABULATE - TABLE - STYLE (2)	
	PROC TABULATE - TITLE	
PROC TRANSPOSE	PROC TRANSPOSE (1)	
	PROC TRANSPOSE (2)	
	PROC TRANSPOSE - SingleCurrency Transactions	
PROC TRANSPOSE2	PROC TRANSPOSE2 - MultiCurrency Transactions	
PROC UPDATE	PROC UPDATE - Merge two cubes	
STYLESHEET	STYLESHEET - PROC TABULATE (1)	
	STYLESHEET - PROC TABULATE (2)	
XML Definition Schema	EF_LOOPEF_VALUEEF_ATTRIBEF_GEF_MACRO	
	EF_SELECTEF_CASEEF_OTHERWISE	

## 5.1. Examples - DATA step

The following examples illustrate the DATA step, subordinate statements and controls:

- [DATA - ARRAY](#)
- [DATA - COLUMN - OUTPUT](#)
- [DATA - DO WHILE](#)
- [DATA - DO WHILE - CONTINUE](#)
- [DATA - DO WHILE - PUT](#)
- [DATA - DO WHILE - PUT - HTML tags](#)
- [DATA - EXTEND](#)
- [DATA - FOR TO NEXT - OUTPUT](#)
- [DATA - IF THEN ELSE](#)
- [DATA - INFILE - COLUMN](#)
- [DATA - INFILE - INLINE \(1\)](#)
- [DATA - INFILE - INLINE \(2\)](#)
- [DATA - MERGE](#)
- [DATA - MERGE - WHERE](#)
- [DATA - MERGE - IN - IF THEN ELSE - OUTPUT](#)
- [DATA - MERGE - IN - \\_OUTPUT\\_](#)
- [DATA - OUTPUT - %PARAM](#)
- [DATA - RECURSE - IF THEN ELSE](#)
- [DATA - RECURSE - IF THEN ELSE - \\_LEVEL\\_](#)
- [DATA - SET - SELECT WHEN THEN](#)
- [DATA - SET - WHERE](#)
- [DATA - SET - IN](#)
- [DATA - SET - DCE.datatable](#)
- [DATA - SQL \(1\)](#)
- [DATA - SQL \(2\)](#)

### 5.1.1. DATA - ARRAY

#### 5.1.1.1. Goal

We start out with a table that lists transactions per fund and investor, and multiplies the lines according to the number of instruments being involved in the transaction.

Fund Id	Investor Id	Share index	Share name	Share price
102	324	1	A1	1
102	324	2	A2	20
102	324	3	A3	30
102	324	4	A4	40
103	524	1	A1	1
103	524	2	A2	20
103	524	3	A3	30
103	524	4	A4	40

We want to transform this table, so as to have only one single line per transaction, regardless of the number of instruments being involved.

Fund Id	Investor Id	Share name of type 1	Share name of type 2	Share name of type 3	Share name of type 4	Share price of type 1	Share price of type 2	Share price of type 3	Share price of type 4
102	324	A1	A2	A3	A4	1	20	30	40
103	524	A1	A2	A3	A4	1	20	30	40

### 5.1.1.2. Features being illustrated

- DATA step
- ARRAY statement
- PROC MEANS step
- LAST statement

### 5.1.1.3. Program

//Create an alias that points to the existing table location

```
LIBNAME TEST "\{Private}\TEST\EXAMPLES\TablesFormation";
```

//Print the input table

```
PROC PRINT DATA=TEST.INPUT_TRANSACTIONS NOOBS LABEL;
```

RUN;

```
/*Create a new table by copying the input table and adding 2 group of columns that
function as array elements for 2 input table columns.*/

DATA TEST.OUTPUT_SHARES (DROP = SHARE_INDEX SHARE_NAME
SHARE_PRICE);

SET TEST.INPUT_TRANSACTIONS;

COLUMN SHARE_NAME_1 LABEL="Share name of type 1" TYPE=STRING;
COLUMN SHARE_NAME_2 LABEL="Share name of type 2" TYPE=STRING;
COLUMN SHARE_NAME_3 LABEL="Share name of type 3" TYPE=STRING;
COLUMN SHARE_NAME_4 LABEL="Share name of type 4" TYPE=STRING;
COLUMN SHARE_PRICE_1 LABEL="Share price of type 1" TYPE=INTEGER;
COLUMN SHARE_PRICE_2 LABEL="Share price of type 2" TYPE=INTEGER;
COLUMN SHARE_PRICE_3 LABEL="Share price of type 3" TYPE=INTEGER;
COLUMN SHARE_PRICE_4 LABEL="Share price of type 4" TYPE=INTEGER;

//Transform the columns SHARE_NAME and SHARE_PRICE into arrays, and define for
array elements for each array.

ARRAY SHARE_NAME{} SHARE_NAME_1-SHARE_NAME_4;
ARRAY SHARE_PRICE{} SHARE_PRICE_1-SHARE_PRICE_4;

//Migrate the values from the columns SHARE_NAME and SHARE_PRICE into the
corresponding array elements.

IF (SHARE_INDEX>=1 AND SHARE_INDEX<=4) THEN

SHARE_NAME{SHARE_INDEX} = SHARE_NAME;
SHARE_PRICE{SHARE_INDEX} = SHARE_PRICE;

END;
```

```
RUN;
```

```
//Display the result
```

```
PROC PRINT DATA=TEST.OUTPUT_SHARES LABEL NOOBS;
```

```
RUN;
```

```
//Regroup columns
```

```
PROC MEANS DATA=TEST.OUTPUT_SHARES;
```

```
CLASS FUND_ID INVESTOR_ID;
```

```
LAST SHARE_NAME_1 (LABEL="Share name of type 1")
```

```
SHARE_NAME_2 (LABEL="Share name of type 2")
```

```
SHARE_NAME_3 (LABEL="Share name of type 3")
```

```
SHARE_NAME_4 (LABEL="Share name of type 4");
```

```
LAST SHARE_PRICE_1 (LABEL="Share price of type 1")
```

```
SHARE_PRICE_2 (LABEL="Share price of type 2")
```

```
SHARE_PRICE_3 (LABEL="Share price of type 3")
```

```
SHARE_PRICE_4 (LABEL="Share price of type 4");
```

```
RUN;
```

```
//Display the result
```

```
PROC PRINT DATA=TEST.OUTPUT_SHARES LABEL NOOBS;
```

```
RUN;
```

## 5.1.1.4. Result

Fund Id	Investor Id	Share index	Share name	Share price					
102	324		1 A1		1				
102	324		2 A2		20				
102	324		3 A3		30				
102	324		4 A4		40				
103	524		1 A1		1				
103	524		2 A2		20				
103	524		3 A3		30				
103	524		4 A4		40				
Fund Id	Investor Id	Share name of type 1	Share name of type 2	Share name of type 3	Share name of type 4	Share price of type 1	Share price of type 2	Share price of type 3	Share price of type 4
102	324	A1				1			
102	324		A2				20		
102	324			A3				30	
102	324				A4				
103	524	A1				1			
103	524		A2				20		
103	524			A3				30	
103	524				A4				
Fund Id	Investor Id	Share name of type 1	Share name of type 2	Share name of type 3	Share name of type 4	Share price of type 1	Share price of type 2	Share price of type 3	Share price of type 4
102	324	A1	A2	A3	A4	1	20	30	
103	524	A1	A2	A3	A4	1	20	30	

## 5.1.2. DATA - COLUMN - OUTPUT

### 5.1.2.1. Goal

Create a table using the COLUMN statement.

### 5.1.2.2. Features being illustrated

- DATA step
- COLUMN statement
- OUTPUT statement

### 5.1.2.3. Program

```
DATA TableSimple;
```

```
COLUMN Id;
```

```
COLUMN FirstName;
```

```
COLUMN Name;
```

```
Id = "101";
```

```
FirstName= "Carol";
```

```
Name = "Smith";
```

```
OUTPUT;
```

```
Id = "102";
```

```
FirstName = "Peter";
```

```
Name = "Morgan";
```

```
OUTPUT;
```

```
RUN;
```

```
PROC PRINT DATA = TableSimple;
```

```
RUN;
```

## 5.1.2.4. Result

	ID	FIRSTNAME	NAME
1	101	Carol	Smith
2	102	Peter	Morgan

## 5.1.3. DATA - DO WHILE

### 5.1.3.1. Goal

Create a table using a DO WHILE control.

### 5.1.3.2. Features being illustrated

- DATA step
- DO WHILE control
- function applied to a character string

### 5.1.3.3. Program

//Create an alias

```
LIBNAME TEST "\EFront_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a table

```
DATA ROBERT_DO;
```

```
COLUMN R TYPE=CHARLABEL="Name";
```

```
COLUMN I TYPE=INTEGER;
```

```
R="Robert";
```

```
I=0;
```

//Apply the DO loop

```
DO WHILE(I<5)
```

```
R=MID(R,2,5);  
I=I+1;  
OUTPUT;  
LOOP;  
RUN;  
//Print the table  
PROC PRINT DATA=ROBERT_DO (DROP=I);  
RUN;
```

#### 5.1.3.4. Result

	R	I
1	obert	5
2	bert	4
3	ert	3
4	rt	2
5	t	1
6		0

### 5.1.4. DATA - DO WHILE - CONTINUE

#### 5.1.4.1. Goal

Exit a control using the CONTINUE statement.

#### 5.1.4.2. Features being illustrated

- DATA step
- CONTINUE statement
- DO WHILE control

### 5.14.3. Program

```
//Create an alias

LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";

//Create a table

DATA ROBERT_DO;

COLUMN R TYPE=CHARLABEL="Name";
COLUMN I TYPE=INTEGER;

R="Robert";
I=LENGTH(R);

//Apply the DO loop

DO WHILE(I<5)

R=MID(R,2,5);

I=I+1;

OUTPUT;

IF (I="4") THEN

CONTINUE;

END;

LOOP;

RUN;

//Print the table

PROC PRINT DATA=ROBERT_DO;
```

RUN;

#### 5.1.4.4. Result

	R	I
1	obert	5
2	bert	4

### 5.1.5. DATA - DO WHILE - PUT

#### 5.1.5.1. Goal

Display information using the PUT statement.

#### 5.1.5.2. Features being illustrated

- DATA step
- PUT statement
- DO...WHILE control

#### 5.1.5.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a table

```
DATA ROBERT_DO;
```

```
COLUMN R TYPE=CHARLABEL="Name";
```

```
COLUMN I TYPE=INTEGER;
```

```
R="Robert";
```

```
I=LENGTH(R);
```

//Apply the DO loop

```
DO WHILE(I>0)
R=MID(R,2,5);
I=I-1;
OUTPUT;
PUT "I = "|I|"; ";
LOOP;
RUN;
//Print the table
PROC PRINT DATA=ROBERT_DO;
RUN;
```

#### 5.1.5.4. Result

I= 5; I= 4; I= 3; I= 2; I= 1; I= 0;		
	R	I
1	obert	5
2	bert	4
3	ert	3
4	rt	2
5	t	1
6		0

#### 5.1.6. DATA - DO WHILE - PUT - HTML tags

##### 5.1.6.1. Goal

Style the PUT statement output using HTML tags

### 5.1.6.2. Features being illustrated

- DATA step
- PUT statement
- DO...WHILE control
- inclusion of HTML tags

### 5.1.6.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a table

```
DATA ROBERT_DO;
```

```
COLUMN R TYPE=CHARLABEL="Name";
```

```
COLUMN I TYPE=INTEGER;
```

```
R="Robert";
```

```
I=LENGTH(R);
```

//Apply the DO loop

```
DO WHILE(I>0)
```

```
R=MID(R,2,5);
```

```
I=I-1;
```

```
OUTPUT;
```

```
PUT "I = "|| I ||"; ";
```

```
PUT "<br>";
```

```
LOOP;
```

RUN;

//Print the table

PROC PRINT DATA=ROBERT\_DO;

RUN;

#### 5.1.6.4. Result

```
I=5;  
I=4;  
I=3;  
I=2;  
I=1;  
I=0;
```

	R	I
1	obert	5
2	bert	4
3	ert	3
4	rt	2
5	t	1
6		0

#### 5.1.7. DATA - EXTEND

##### 5.1.7.1. Goal

...

##### 5.1.7.2. Features being illustrated

- DATA step
- EXTEND statement
- Functions - special

##### 5.1.7.3. Program

```
/*...*/
```

```
DATA WORK.T_Investment_structure (DROP=COMMITMENT_FUND  
TOTAL_COMMITMENT_FUND);  
  
EXTEND WORK.T_Investment_structure (KEEP=COMMITMENT_FUND  
TOTAL_COMMITMENT_FUND);  
  
COLUMN STAKE TYPE=FLOAT;  
  
If NZ(TOTAL_COMMITMENT_FUND,0) <> 0 THEN  
  
STAKE = COMMITMENT_FUND / TOTAL_COMMITMENT_FUND;  
  
END;  
  
STAKE = NZ(STAKE,1);  
  
RUN;
```

#### 5.1.7.4. Result

### 5.1.8. DATA - FOR IN NEXT - %PARAM

#### 5.1.8.1. Goal

Sort a data table according to the order of an ??XID selection.

#### 5.1.8.2. Features being illustrated

- [DATA step](#)
- [FOR...IN...NEXT controls](#)
- [%PARAM statement](#)

#### 5.1.8.3. Program

//Create an alias

```
LIBNAME USER ".":
```

```
%PARAM CONTACTS INFORMAT="FUNDS" INFORMAT="??XID(SFACONTACT)":
```

//Prepare a table with ID and INDEX reflecting the order of the ??XID selection.

```
DATA WORK.TEMP;  
COLUMN ID;  
COLUMN INDEX TYPE=INTEGER;  
//Instantiate INDEX  
INDEX=0;  
//Apply the FOR control  
FOR ID IN (%CONTACTS)  
OUTPUT;  
INDEX=INDEX+1;  
NEXT;  
RUN;  
//Filter the view  
DATA T_CONTACTS;  
SET CONTACTS (WHERE ID IN(%CONTACTS));  
RUN;  
//Merge both tables to insert the INDEX column in the target table  
DATA T_CONTACTS;  
MERGE T_CONTACTS WORK.TEMP;  
BY ID;  
RUN;  
//Sort the table by INDEX, eventually drop the INDEX column afterwards
```

```
PROC SORT DATA = T_CONTACTS  
OUT = T_CONTACTS;  
BY INDEX;  
RUN;  
//Print the table  
PROC PRINT DATA=T_CONTACTS;  
RUN;
```

#### 5.1.8.4. Result

### 5.1.9. DATA - FOR TO NEXT - OUTPUT

#### 5.1.9.1. Goal

Create a table using a FOR control.

#### 5.1.9.2. Features being illustrated

- DATA step
- FOR...TO...NEXT control
- OUTPUT statement

#### 5.1.9.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a table

```
DATA TEST.ABCD_FOR;
```

//Create the columns of the table

```
COLUMN A TYPE=INTEGER;
```

```
COLUMN B TYPE=INTEGER;  
COLUMN C TYPE=INTEGER;  
COLUMN D TYPE=INTEGER;  
//Apply the FOR group  
FOR A=1 TO 2  
FOR B=1 TO 2  
FOR C=1 TO 2  
D=A*B*C;  
OUTPUT;  
NEXT;  
NEXT;  
NEXT;  
RUN;  
//Print the table  
PROC PRINT DATA=TEST.ABCD_FOR;  
RUN;
```

#### 5.1.9.4. Result

	A	B	C	D
1	1	1	1	1
2	1	1	2	2
3	1	2	1	2
4	1	2	2	4
5	2	1	1	2
6	2	1	2	4
7	2	2	1	4
8	2	2	2	8

## 5.1.10. DATA - INFILE - COLUMN

### 5.1.10.1. Goal

Import data from an XLS file on the server.

### 5.1.10.2. Features being illustrated

- [DATA step](#)
- [INFILE statement](#)
- [COLUMN statement](#)

### 5.1.10.3. Program

```
DATA HF;  
  
INFILE "C:\TMP\HF.XLS" FIRSTOBS=2;  
  
COLUMN Fund_Name LABEL="Name";  
  
COLUMN Fund_Domicile LABEL="Domicile ";  
  
COLUMN Fund_Type LABEL="Type";  
  
COLUMN Fund_Status LABEL="Status ";  
  
COLUMN Fund_AUM_USD LABEL="AUM(USD)" TYPE=FLOAT;  
  
COLUMN Fund_Currency LABEL="Currency";  
  
COLUMN Fund_Inception_Date LABEL="Inception" TYPE=DATE;  
  
COLUMN Fund_Reporting_Style LABEL="Reporting";  
  
COLUMN Fund_Management_Fee LABEL="Management fee" TYPE=FLOAT;  
  
COLUMN Fund_Performance_Fee LABEL="Performance fee" TYPE=FLOAT;  
  
COLUMN Fund_Highwater_Mark LABEL="Highwater Mark" TYPE=BOOLEAN;  
  
COLUMN Fund_Leverage LABEL="Leverage" TYPE=FLOAT;
```

```
COLUMN Fund_Investment_Minimum LABEL="Inv.Minimum" TYPE=FLOAT;  
COLUMN Fund_Subscription_Frequency LABEL="Subscription freq";  
COLUMN Fund_Lockup_Period LABEL="Lockup period";  
COLUMN Fund_Redemption_Frequency LABEL="Redemption freq";  
COLUMN Fund_Strategy_Classif_1 LABEL="Strategy";  
COLUMN Fund_Geographical_Focus LABEL="Geography";  
COLUMN Fund_Primary_Benchmark LABEL="Benchmark";  
COLUMN Fund_Investor_Type LABEL="Investor type";  
COLUMN Fund_NAV LABEL="NAV" TYPE=FLOAT;  
RUN;  
PROC PRINT DATA = HF;  
RUN;
```

## 5.1.11. DATA - INFILE - INLINE (2)

### 5.1.11.1. Goal

Create a table that states parent-child relationships.

### 5.1.11.2. Features being illustrated

- DATA step
- INFILE statement
- INLINE statement
- COLUMN statement

### 5.1.11.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
//Create a table expliciting parent/child links  
DATA TEST.GENERATIONS;  
INFILE INLINE;  
COLUMN FATHER LABEL="Father";  
COLUMN CHILD LABEL="Child";  
INLINE;  
FATHER="" CHILD="Smith"  
FATHER="Smith" CHILD="Smith_1"  
FATHER="Smith" CHILD="Smith_2"  
FATHER="Smith" CHILD="Smith_3"  
CHILD="Brown" FATHER=""  
FATHER="Brown" CHILD="Brown_1"  
FATHER="Brown" CHILD="Brown_2"  
FATHER="Brown" CHILD="Brown_3"  
FATHER="Brown_1" CHILD="Brown_11"  
FATHER="Brown_1" CHILD="Brown_12"  
FATHER="Brown_2" CHILD="Brown_21"  
FATHER="Brown_2" CHILD="Brown_22"  
FATHER="Brown_2" CHILD="Brown_23"  
;
```

RUN;

//Print the table

PROC PRINT DATA=TEST.GENERATIONS NOOBS LABEL;

RUN;

#### 5.1.11.4. Result

Father	Child
	Smith
Smith	Smith_1
Smith	Smith_2
Smith	Smith_3
	Brown
Brown	Brown_1
Brown	Brown_2
Brown	Brown_3
Brown_1	Brown_11
Brown_1	Brown_12
Brown_2	Brown_21
Brown_2	Brown_22
Brown_2	Brown_23

#### 5.1.12. DATA - INFILE - INLINE (1)

##### 5.1.12.1. Goal

Create a table using INFILE and INLINE statements.

##### 5.1.12.2. Features being illustrated

- DATA step
- INFILE statement
- INLINE statement

##### 5.1.12.3. Program

//Create the alias that points the existing table location

LIBNAME TEST "\\\EFront\_SUP.2/Private\\TEST\\EXAMPLES\\TablesFormation";

```
//Create a table that contains a user record  
  
DATA TEST.User1;  
  
//Create the table structure  
  
INFILE INLINE;  
  
COLUMN USER_ID LABEL="Id" TYPE=INTEGER;  
  
COLUMN FIRSTNAME LABEL="First Name";  
  
COLUMN LASTNAME LABEL="Name";  
  
COLUMN BIRTHDAY LABEL="Birthday" TYPE=DATE;  
  
COLUMN ZIP LABEL="Zip Code" TYPE=INTEGER;  
  
COLUMN CITY LABEL="City";  
  
//Add values  
  
INLINE;  
  
201,"Elisabeth","Montgomery","04/01/1983","78000","Versailles";  
  
RUN;  
  
//Display data on screen  
  
PROC PRINT DATA = TEST.User1;  
  
RUN;
```

#### 5.1.12.4. Result

	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	ZIP	CITY
1	201	Elisabeth	Montgomery	04/01/1983	78000	Versailles

## 5.1.13. DATA - IF THEN ELSE

### 5.1.13.1. Goal

According to the value of the amount and value of contracts per user, qualify the type of contract.

### 5.1.13.2. Features being illustrated

- DATA step
- IF...THEN...ELSE control

### 5.1.13.3. Program

```
//Build result table
DATA TEST.CONTRACT_PER_USER;
MERGE TEST.USERS_INFO_ABOUT_CONTRACTS (DROP = CONTRACT_ID UNIT_PRICE QTY) WORK.RESULTS;
BY USER_ID;

//Add new column
COLUMN TYPE;

//Calculate cell values
IF AMOUNT < 10000 THEN
TYPE = "Small";
ELSEIf AMOUNT < 20000 THEN
TYPE = "Average";
ELSE
TYPE = "Big";
END;
RUN;

//Display the result
PROC PRINT DATA = TEST.CONTRACT_PER_USER;
RUN;
```

### 5.1.13.4. Result

	USER_ID	FIRSTNAME	LASTNAME	CHILDREN	N_CONTRAC	AMOUNT	AVERAGE	TYPE
1	101	Bobby	Schmidt	2	4	19450,3	2,75	Average
2	102	Ruber	Zodi	1	2	6441,6	50,5	Small
3	103	Deborah	Woodpecker	5	1	7501	3	Small
4	105	Marjorie	Pattern	0	1	20030,2	5	Big

## 5.1.14. DATA - MERGE

### 5.1.14.1. Goal

Merge two tables defining one common variable

### 5.1.14.2. Features being illustrated

- [DATA step](#)
- [MERGE statement](#)

### 5.1.14.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TableFormation";
```

//Define the output table

```
DATA TEST.USERS_AND_CONTRACTS;
```

//Define the input tables

```
MERGE TEST.USERS TEST.CONTRACTS;
```

//Define the common variable

```
BY USER_ID;
```

```
RUN;
```

//Print the table

```
PROC PRINT DATA=TEST.USERS_AND_CONTRACTS;
```

```
RUN;
```

## 5.1.14.4. Result

	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	ZIP	CITY	CONTRACT	UNIT_PRICE	QTY
1	101	Bobby	Schmidt	05/03/1959	10021	New York	593	4220,1	2
2	101	Bobby	Schmidt	05/03/1959	10021	New York	594	1299	3
3	101	Bobby	Schmidt	05/03/1959	10021	New York	595	7501	1
4	101	Bobby	Schmidt	05/03/1959	10021	New York	597	6430,2	5
5	102	Ruber	Zodi	25/01/1980	75001	Paris	596	11,4	100
6	102	Ruber	Zodi	25/01/1980	75001	Paris	597	6430,2	1
7	103	Sandra	Specker	01/12/1962	33000	Bordeaux	596	7501	3
8	104	Sandra	Specker	01/12/1962	33000	Bordeaux			
9	105	Marjorie	Pattern	12/03/1983	10021	New York			

## 5.1.15. DATA - MERGE - WHERE

### 5.1.15.1. Goal

Merge two tables with a condition applied to the common variable

### 5.1.15.2. Features being illustrated

- DATA step
- MERGE statement
- WHERE table option

### 5.1.15.3. Program

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

```
//Define the output table
```

```
DATA TEST.USERS_AND_CONTRACTS;
```

```
//Define the input tables
```

```
MERGE TEST.USERS TEST.CONTRACTS (WHERE(USER_ID="101"));
```

```
//Define the common variable
```

```
BY USER_ID;
```

```
RUN;
```

//Print the table

```
PROC PRINT DATA=TEST.USERS_AND_CONTRACTS;
RUN;
```

### 5.1.15.4. Result

	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	CHILDREN	ZIP	CITY	CONTRACT	UNIT_PRICE	QTY
1	101	Bobby	Schmidt	05/03/1959	2	10021	New York	593	4220,1	2
2	101	Bobby	Schmidt	05/03/1959	2	10021	New York	594	1299	3
3	101	Bobby	Schmidt	05/03/1959	2	10021	New York	595	7501	1
4	101	Bobby	Schmidt	05/03/1959	2	10021	New York	599	6430,2	5
5	102	Ruber	Zodi	25/01/1980	1	75001	Paris			
6	103	Deborah	Woodpecker	04/11/1954	5	33000	Bordeaux			
7	104	Sandra	Specker	01/12/1962	4	33000	Bordeaux			
8	105	Marjorie	Pattern	12/03/1983		10021	New York			

### 5.1.15.5. Comment

- Notice that all the lines of the table mentioned first in the MERGE statement are included in the merge result. The WHERE filter applies only to the lines of the second table mentioned in the MERGE statement. Therefore no values are listed for columns CONTRACT\_ID, UNIT\_PRICE and QTY. If the tables mentioned in the MERGE statement change position, the merge result is different.

```
MERGE TEST.CONTRACTS TEST.USERS (WHERE(USER_ID="101"));
```

	CONTRACT	USER_ID	UNIT_PRICE	QTY	FIRSTNAME	LASTNAME	BIRTHDAY	CHILDREN	ZIP	CITY
1	593	101	4220,1	2	Bobby	Schmidt	05/03/1959	2	10021	New York
2	594	101	1299	3	Bobby	Schmidt	05/03/1959	2	10021	New York
3	595	101	7501	1	Bobby	Schmidt	05/03/1959	2	10021	New York
4	599	101	6430,2	5	Bobby	Schmidt	05/03/1959	2	10021	New York
5	596	102	11,4	100						
6	597	102	6430,2	1						
7	598	103	7501	3						
8	600	105	20030,2	5						

### 5.1.16. DATA - MERGE - IN - IF THEN ELSE - OUTPUT

#### 5.1.16.1. Goal

Merge two tables and select lines using IN.

### 5.1.16.2. Features being illustrated

- DATA step
- MERGE statement
- IN table option
- IF...THEN...ELSE control
- OUTPUT statement

### 5.1.16.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Define the output table

```
DATA TEST.USERS_AND_CONTRACTS;
```

//Define the input tables

```
MERGE TEST.USERS (IN=is_customer) TEST.CONTRACTS(IN=has_contract);
```

//Define the common variable

```
BY USER_ID;
```

//Apply the condition

```
If is_customer AND has_contract
```

```
THEN OUTPUT;
```

```
END;
```

```
RUN;
```

//Print the table

```
PROC PRINT DATA=TEST.USERS_AND_CONTRACTS;
```

```
RUN;
```

## 5.1.16.4. Result

	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	CHILDREN	ZIP	CITY	CONTRACT	UNIT_PRICE	QTY
1	101	Bobby	Schmidt	05/03/1959	2	10021	New York	593	4220,1	2
2	101	Bobby	Schmidt	05/03/1959	2	10021	New York	594	1299	3
3	101	Bobby	Schmidt	05/03/1959	2	10021	New York	595	7501	1
4	101	Bobby	Schmidt	05/03/1959	2	10021	New York	599	6430,2	5
5	102	Ruber	Zodi	25/01/1980	1	75001	Paris	596	11,4	100
6	102	Ruber	Zodi	25/01/1980	1	75001	Paris	597	6430,2	1
7	103	Deborah	Woodpecker	04/11/1954	5	33000	Bordeaux	598	7501	3
8	105	Marjorie	Pattern	12/03/1983		10021	New York	600	20030,2	5

## 5.1.16.5. Comment

- The result includes only data lines with common values for USER\_ID in both tables.

## 5.1.17. DATA - MERGE - IN - \_OUTPUT\_

### 5.1.17.1. Goal

Limit the amount of data when merging two tables.

### 5.1.17.2. Features being illustrated

- DATA step
- MERGE statement
- IN flag
- \_OUTPUT\_flag

### 5.1.17.3. Program

```
DATA [SP_RP_TB].[EFFV_CRP_T_ACCOUNT_TO_CREDIT];
```

```
MERGE [WORK].[SOUSCRIPTOR] [SP_RP_TB].  
[EFFV_CRP_T_ACCOUNT_TO_CREDIT](IN=T1);
```

```
BY INVESTOR_ID;
```

```
_OUTPUT_=T1;
```

```
RUN;
```

## 5.1.17.4. Comment

- The data line is only written to the output table, if it meets the condition expressed using the IN flag together with the \_OUTPUT\_ flag.

## 5.1.18. DATA - OUTPUT - %PARAM

### 5.1.18.1. Goal

Add data lines to an existing table.

### 5.1.18.2. Features being illustrated

- DATA step
- OUTPUT statement
- %PARAM statement

### 5.1.18.3. Program

```
%PARAM PARAM_NAME1 TYPE=STRING;  
  
%PARAM PARAM_NAME2 TYPE=STRING;  
  
DATA work.tmp;  
  
COLUMN COL1 TYPE=STRING;  
  
COLUMN NAME TYPE=STRING;  
  
COL1 = "First line";  
  
NAME = %PARAM_NAME1;  
  
OUTPUT; // Write a line in the table  
  
COL1 = "Second line";  
  
NAME = %PARAM_NAME2;  
  
OUTPUT; // Write another line in the table
```

```
RUN;  
PROC PRINT DATA = work.tmp;  
RUN;
```

## 5.1.19. DATA - RECURSE - IF THEN ELSE

### 5.1.19.1. Goal

Calculate parent-child links from an [existing table](#)

### 5.1.19.2. Features being illustrated

- DATA step
- RECURSE statement
- IF...THEN...ELSE control

### 5.1.19.3. Program

```
//Create an alias  
  
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
  
//Calculate existing links between father and child from an existing table  
  
DATA TEST.GENERATIONS_TREE;  
  
RECURSE TEST.GENERATIONS (VAR=CHILD PARENT=FATHER START=(FATHER  
IS NULL));  
  
COLUMN PARENTHOOD LABEL="Parent-child relation";  
  
//Apply the condition  
  
IF FATHER IS NULLTHEN  
  
PARENTHOOD=CHILD;  
  
ELSE  
  
PARENTHOOD=PARENT (PARENTHOOD)&"\ "&CHILD;
```

```

END;

RUN;

//Print the table

PROC PRINT DATA=TEST.GENERATIONS_TREE NOOBS LABEL;

VAR CHILD (LABEL="Person") PARENTHOOD;

RUN;

```

## 5.1.19.4. Result

Person	Parent-child relation
Smith	Smith
Smith_1	Smith: Smith_1
Smith_2	Smith: Smith_2
Smith_3	Smith: Smith_3
Brown	Brown
Brown_1	Brown: Brown_1
Brown_11	Brown: Brown_1\ Brown_11
Brown_12	Brown: Brown_1\ Brown_12
Brown_2	Brown: Brown_2
Brown_21	Brown: Brown_2\ Brown_21

## 5.1.20. DATA - RECURSE - IF THEN ELSE - \_LEVEL\_

### 5.1.20.1. Goal

Calculate parent-child links indicating hierarchical levels.

### 5.1.20.2. Features being illustrated

- DATA step
- RECURSE statement
- \_LEVEL\_ variable

### 5.1.20.3. Program

```
//Create an alias

LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";

//Calculate existing links between father and child from an existing table

DATA TEST.GENERATIONS_TREE;

RECURSE TEST.GENERATIONS (VAR=CHILD PARENT=FATHER START=(FATHER
IS NULL));

COLUMN LEVEL;

COLUMN PARENTHOOD LABEL="Parent-child relation";

LEVEL=_LEVEL_;

//Apply the condition

IF FATHER IS NULLTHEN

PARENTHOOD=CHILD;

ELSE

PARENTHOOD=PARENT (PARENTHOOD)&"\ "&CHILD;

END;

RUN;

//Print the table

PROC PRINT DATA=TEST.GENERATIONS_TREE NOOBS LABEL;

VAR LEVEL CHILD (LABEL="Person") PARENTHOOD;

RUN;
```

## 5.1.20.4. Result

LEVEL	Person	Parent-child relation
0	Smith	Smith
1	Smith_1	Smith: Smith_1
1	Smith_2	Smith: Smith_2
1	Smith_3	Smith: Smith_3
0	Brown	Brown
1	Brown_1	Brown: Brown_1
2	Brown_11	Brown: Brown_1\ Brown_11
2	Brown_12	Brown: Brown_1\ Brown_12
1	Brown_2	Brown: Brown_2
2	Brown_21	Brown: Brown_2\ Brown_21

## 5.1.21. DATA - SET - WHERE

### 5.1.21.1. Goal

Filter an input table using SET and WHERE.

### 5.1.21.2. Features being illustrated

- DATA step
- SET statement
- WHERE table option

### 5.1.21.3. Program

//Create an alias

```
LIBNAME TEST "\EFront_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Define the output table

```
DATA TEST.USERS_1980;
```

//Read a selection of lines from the input table

```
SET TEST.USERS (WHERE YEAR(BIRTHDAY)>= 1980);
```

```
RUN;
```

```
//Print the table
```

```
PROC PRINT DATA=TEST.USERS_1980;
```

```
RUN;
```

#### 5.1.21.4. Result

	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	ZIP	CITY
1	102	Ruber	Zedi	25/01/1980	75001	Paris
2	105	Marjorie	Pattern	12/03/1983	10021	New York

### 5.1.22. DATA - SET - SELECT WHEN THEN

#### 5.1.22.1. Goal

Create column values using a SELECT group

#### 5.1.22.2. Features being illustrated

- [DATA step](#)
- [SET statement](#)
- [SELECT...WHEN...THEN control](#)

#### 5.1.22.3. Program

```
//Create an alias that points to the existing table location
```

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

```
/*Add a column whose values are calculated using a
```

```
SELECT group to an existing table*/
```

```
DATA TEST.CUSTOMERS_CHILDREN (KEEP=FIRSTNAME LASTNAME CHILDREN
TOYS);
```

```
//Read the input table  
  
SET TEST.USERS;  
  
COLUMN TOYS TYPE=CHAR;  
  
//Apply SELECT...WHEN control  
  
SELECT (CHILDREN)  
  
WHEN (1) THEN  
  
TOYS="Send 1 toy";  
  
END;  
  
WHEN (2) THEN  
  
TOYS="Send 2 toys";  
  
END;  
  
OTHERWISE  
  
TOYS="No toys";  
  
END;  
  
END;  
  
RUN;  
  
//Print the table  
  
PROC PRINT DATA=TEST.CUSTOMERS_AND_CHILDREN;  
  
RUN;
```

## 5.1.22.4. Result

	FIRSTNAME	LASTNAME	CHILDREN	TOYS
1	Bobby	Schmidt	2	Send 2 toys
2	Ruber	Zodi	1	Send 1 toy
3	Deborah	Woodpecker	5	No toys
4	Sandra	Specker	4	No toys
5	Marjorie	Pattern	0	No toys

## 5.1.23. DATA - SET - IN

### 5.1.23.1. Goal

When merging two tables, identify the table from which a line stems from, and make it undergo a special treatment. We start out with a table that shows investor positions for **direct investment**, and a table that shows investor positions for **fund investment**.

Fund investor	Fund	Commitment
Fund investor A	Fund A	125000
Fund investor B	Fund A	325000
Fund investor C	Fund A	525000

Portfolio investor	Fund	Commitment
Portfolio investor A	Fund A	135000
Portfolio investor B	Fund A	435000

If the value for **Commitment** stems from the Portfolio investor table, we want to write the value to a new column, called **Portfolio investment**.

### 5.1.23.2. Features being illustrated

- DATA step
- SET statement
- Functions - special

### 5.1.23.3. Program

//Create an alias that points to the existing table location

```
LIBNAME TEST "\{Private\}\TEST\EXAMPLES\TablesFormation";  
/*Merge the input tables, and write the commitment value of the Portfolio investor table  
into a new column*/  
  
DATA TEST.INVESTOR_POSITIONS;  
  
SET TEST.FUND_INVESTORS(IN=Fund)  
TEST.PORTFOLIO_INVESTORS(IN=Portfolio);  
  
COLUMN PORTFOLIO_INVESTMENT LABEL = "Portfolio investment" TYPE=FLOAT;  
  
If Portfolio THEN  
  
PORTFOLIO_INVESTMENT = NZ(COMMITMENT,0);("", "  
END;  
  
RUN;  
  
//Display the result  
  
PROC PRINT DATA = TEST.INVESTOR_POSITIONS NOOBS LABEL;  
  
RUN;
```

#### 5.1.23.4. Result

Fund investor	Fund	Commitment	Portfolio investment
Fund investor A	Fund A	125000	
Fund investor B	Fund A	325000	
Fund investor C	Fund A	525000	
Portfolio investor A	Fund A	135000	135000
Portfolio investor B	Fund A	435000	435000

## 5.1.24. DATA - SET - FCE.datatable

### 5.1.24.1. Goal

Call the **eFront Invest Funds calculation** engine from within an eFront Script program.

### 5.1.24.2. Features being illustrated

- eFront Report libraries
- %DEFINE statement
- %LET statement
- Functions - special
- SET statement

### 5.1.24.3. Program

```
LIBNAME USER ":";  
  
%DEFINE REPORT_DATE;  
  
%LET DATE=TODAY();  
  
DATA WORK.list;  
  
SQL "select IQID from vcfund A where ??FILTER";  
  
COLUMN IQID;  
  
RUN;  
  
DATA WORK.list;  
  
SET WORK.list(from=1 to=100);  
  
RUN;  
  
%LET iqid_fof_fund = COLLECTION("work.list", "iqid");  
  
//%let iqid_fof_fund = "E8AB57217FD948CE9B44E9BBB4D7AF9E"; // A11
```

```
%DEFINE FOF_INVESTOR_REPORT;  
DATA WORK.tmp1;  
SET FCE.FUNDOPERATIONS;  
RUN;
```

## 5.1.25. DATA - SQL (1)

### 5.1.25.1. Goal

Include an **SQL** query in a DATA step.

### 5.1.25.2. Features being illustrated

- **DATA** step
- **SQL** statement

### 5.1.25.3. Program

```
LIBNAME USER ".";  
DATA WORK.T_FUNDS;  
SQL "A.FUND FROM VCFUND A WHERE ??FILTER;";  
COLUMN FUND LABEL="Fund Name{F}Nom du fonds" TYPE="STRING";  
RUN;  
PROC PRINT DATA=SS;  
WORK.T_FUNDS LABEL;  
RUN;
```

### 5.1.25.4. Comment

- ??FILTER applies implicitly to the table used with the **alias** A. For the moment, you cannot use a different alias. ??FILTER filters the table according to the user region.

If you omit the filter, the query completely overpasses the regions and access rights. Therefore, in most cases, it is better to create a view first, and then copy it into a data table.

## 5.1.26. DATA - SQL (2)

### 5.1.26.1. Goal

Use an **SQL** query to create a data table that lists 'other users/groups in charge of a project'.

### 5.1.26.2. Features being illustrated

- **DATA** step
- **SQL** statement

### 5.1.26.3. Program

DATA Work.PROJOWNER;

SQL "SELECT A.PROJECT, B.FIRSTNAME FROM VCPROJOWNER A, ADMUSER B WHERE A.OWNERID = B.IQID";

COLUMN PROJECT\_ID TYPE=STRING WIDTH=350;

COLUMN USER\_FIRSTNAME TYPE=STRING;

RUN;

PROC PRINT DATA=WORK.PROJOWNER;

RUN;

### 5.1.26.4. Result

	PROJECT_ID	USER_FIRST
1	2002038F7208466EBDF255ECBA487B75	Sylvain
2	BFB4C6EA1B2A404598F63673B85E386A	
3	CFE2DA6AE65A45E38456DD5D1972C89D	Sylvain

## 5.2. Examples - EVENT step

The following examples illustrate the EVENT step, and subordinate statements:

- [EVENT PREDATAROW - \\_URL\\_](#)
- [PROC PRINT - EVENT PREDATAROW](#)

### 5.2.1. EVENT PREDATAROW - \_URL\_

#### 5.2.1.1. Goal

Define in an eFront Report page different links for each cell of the output table.

#### 5.2.1.2. Features being illustrated

- [EVENT step](#)
- [The variable: \\_URL\\_](#)

#### 5.2.1.3. Program

//Activate the capability to specify, in an eFront Report page, a differnt link for each cell of the output table

EVENT PREDATAROW

IF NOT (INVESTOR\_FUND\_ID IS NULL) THEN

//Define the value of the URL addresses to the table cells

\_URL\_="TXT\_REPORT.FOLDER=F\_VCFUND01,  
INVESTOR\_FUND\_ID,INVESTOR\_NAME;TXT\_REPORT.FOLDER=F\_VCINVESTMENT,INVESTMENT\_ID";

ELSEIF NOT NOT (INVESTOR\_COMPANY\_ID IS NULL) THEN

//Define the value of the URL addresses to the table cells

\_URL\_="TXT\_REPORT.FOLDER=F\_VCACCOUNT01,  
INVESTOR\_COMPANY\_ID,INVESTOR\_NAME;TXT\_REPORT.FOLDER=F\_VCINVESTMENT,INVESTMENT\_ID";

```
ELSE
```

```
//Define the value of the URL addresses to the entire table  
  
_URL_="TXT_REPORT.FOLDER=F_VCINVESTMENT,INVESTMENT_ID";  
  
END;
```

#### 5.2.1.4. Comment

By default, we have a link to the investment, and:

- If the investor is a fund, we link to the fund from the name of the fund
- If the investor is a company, we link to the company from the name of the company

Notice that more than one link has been defined for the \_URL\_ variable! You can concatenate define more than one destination links for a table cell when defining the value of the \_URL\_ variable! This allows you as in the example above to define for each cell of a table different links, one which points to the investor, and another one which points to the company.

#### 5.2.1.5. Result

### 5.2.2. PROC PRINT - EVENT PREDATAROW

#### 5.2.2.1. Goal

Print a table, and style table rows according to the value of column content.

#### 5.2.2.2. Features being illustrated

- PROC PRINT step
- EVENT step

#### 5.2.2.3. Program

```
//Display data on screen
```

```
LIBNAME USER ":";
```

```
DATA WORK.T_TEST;
```

```
COLUMN COL1 TYPE=STRING;  
COLUMN AMOUNT TYPE=FLOAT;  
COL1 = "HELLO"; AMOUNT = 20; OUTPUT;  
COL1 = "AAAA"; AMOUNT = 30; OUTPUT;  
COL1 = "BBBB"; AMOUNT = 40; OUTPUT;  
COL1 = "CCCC"; AMOUNT = 50; OUTPUT;  
COL1 = "DDDD"; AMOUNT = 60; OUTPUT;  
RUN;  
  
PROC PRINT DATA=WORK.T_TEST;  
RUN;  
  
PROC PRINT DATA=WORK.T_TEST;  
VAR COL1 AMOUNT;  
EVENT PREDATAROW  
IF AMOUNT > 50 THEN  
    _STYLE_ = "color:#cccccc;";  
ELSEIF AMOUNT > 30 THEN  
    _STYLE_ = "background-color:GREEN;";  
END;  
IF COL1 = "HELLO" THEN  
    _STYLE_ = "background-color:ORANGE;";  
COL1 = "HELLO2";
```

END;

END;

RUN;

#### 5.2.2.4. Result

	COL1	AMOUNT
1	HELLO	20
2	AAAA	30
3	BBBB	40
4	CCCC	50
5	DDDD	60

	COL1	AMOUNT
1	HELLO2	20
2	AAAA	30
3	BBBB	40
4	CCCC	50
5	DDDD	60

## 5.3. Examples - PROC CONVERTCURR step

The following examples illustrate the CONVERTCURR step, and subordinate statements:

- [PROC CONVERTCURR](#)

### 5.3.1. PROC CONVERTCURR

#### 5.3.1.1. Goal

Convert amounts from source cube columns to target cube columns using specified FX rates cube.

#### 5.3.1.2. Features being illustrated

- [PROC CONVERTCURR step](#)
- [NAME option](#)
- [DATA step](#)
- [Functions - date](#)

#### 5.3.1.3. Program

//Create the table from the date filtered cube

DATA WORK.CCY;

SET Universal.CCY\_CONVERSION(WHERE CONVERSIONDATE = YMD(2013, 12, 31));

RUN;

//Create a data column containing the source currency key and the destination currency key

DATA WORK.CCY;

SET WORK.CCY;

COLUMN SRC\_DEST;

```
SRC_DEST=SourceCurrencyKey#"&DestinationCurrencyKey;  
RUN;  
  
//Create data cube  
  
DATA WORK.PORTFOLIOPERFORMANCE;  
  
SET GLOBAL.PORTFOLIOPERFORMANCE;  
  
RUN;  
  
//Create a data column containing the source Transaction currency key and the Fund  
currency key  
  
DATA WORK.PORTFOLIOPERFORMANCE;  
  
SET WORK.PORTFOLIOPERFORMANCE;  
  
COLUMN TRANS_FUND;  
  
TRANS_FUND=TransCurrencyKey#"&FundCurrencyKey;  
  
RUN;  
  
//Create a new column that stores the Remaining commitment to third parties amounts  
converted from the Transaction currency into the Fund currency  
  
PROC CONVERTCURR  
DATA=WORK.PORTFOLIOPERFORMANCE FXRATES=WORK.CCY  
OUT=WORK.PORTFOLIOPERFORMANCE CURRENCY_COLUMN="TRANS_FUND"  
FXRATES_CURRENCY_COLUMN="SRC_DEST"  
FXRATES_RATE_COLUMN="EXCHANGERATE";  
  
VAR REMAININGCOMMITMENTTOTHIRDPARTIES  
(NAME=REMAININGCOMMITMENTTOTHIRDPARTIES_NEW);  
  
RUN;  
  
//Remove the lines where there is no remaining commitment
```

```
DATA WORK.PORTFOLIOPERFORMANCE;  
SET WORK.PORTFOLIOPERFORMANCE(WHERE  
REMAININGCOMMITMENTTOTHIRDPARTIES>0);  
RUN;  
//Print the output cube  
PROC PRINT DATA=WORK.PORTFOLIOPERFORMANCE;  
RUN;
```

### 5.3.1.4. Result

PORTFOLIO	REMAININGCOMMITMENTTOTHIRDPARTIES	IRRITEMSINVESTMENTS
Secondary Investments	23832519.05	23832519.05
Secondary Investments	25766.95	25766.95
Secondary Investments	1066720.64	1066720.64
Secondary Investments	533.36	533.36
Secondary Investments	3924739	3924739
Secondary Investments	106685.57	106685.57
Secondary Investments	106685.57	106685.57
Secondary Investments	42.67	42.67
Secondary Investments	31700.08	31700.08

FUND	CURRENCYKEY	TRANS_FUND	REMAININGCOMMITMENT
			63 189#63
			63 189#63
			189 189#189
			189 189#189
			63 189#63
			63 189#63
			63 189#63
			63 189#63
			189 189#189



## 5.4. Examples - PROC EFRONTACCRUALS step

The following examples illustrate the PROC EFRONTACCRUALS step:

- [PROC EFRONTACCRUALS - %PARAM](#)

### 5.4.1. PROC EFRONTACCRUALS - %PARAM

#### 5.4.1.1. Goal

Generate a table presenting accrued interests for a given date.

#### 5.4.1.2. Features being illustrated

- [PROC EFRONTACCRUALS step](#)
- [%PARAM statement](#)

#### 5.4.1.3. Program

```
%PARAM INS LABEL="Instrument:" INFORMAT="??ID(VCINVESTMENTINS)";

%PARAM DATE LABEL="Date:" TYPE=DATE NOTNULL;

DATA WORK.INSTRUMENTS;

COLUMN INSTRUMENT;

INSTRUMENT=%INS;

OUTPUT;

RUN;

PROC EFRONTACCRUALS DATA=INSTRUMENTS OUT=ACCRUALS
ID=INSTRUMENT USEDRAFT USERECURRING ENDOFDAY DATE=%DATE;

RUN;

PROC PRINT DATA=ACCRUALS;

RUN;
```

### 5.4.1.4. Result

Instrument:	Loan <input type="button" value="X"/>
Date:	30/06/2019 <input type="button" value="Calendar"/>
<b>Apply</b>	
	<input type="button" value="INSTRUMENT"/> <input type="button" value="LOANACCRUALS_PRINCIPAL2"/> <input type="button" value="LOANACCRUALS_WRITEOFF2"/> <input type="button" value="LOANACCRUALS_PIK2"/> <input type="button" value="LOANACCRUALS_BALANCE2"/>
1 FA99A852DA4A...	-400 -400

## 5.5. Examples - PROC EFRONTBLOB step

The following examples illustrate the PROC EFRONTBLOB step:

- PROC EFRONTBLOB - %PARAM
- PROC EFRONTBLOB - DATA - CLASS - PREFIX - ID

### 5.5.1. PROC EFRONTBLOB - %PARAM

#### 5.5.1.1. Goal

Extract data from a blob field (VCSCENARIOTYPE.DEFINITION) table into an eFront Report data table.

#### 5.5.1.2. Features being illustrated

- PROC EFRONTBLOB step
- %PARAM statement

#### 5.5.1.3. Program

```
%PARAM SCENARIO_TYPE_NAME DEFAULT="LP (PI) Portfolio Company";
```

```
DATA WORK.SCENARIO_TYPE;
```

```
SQL "select DEFINITION from VCSCENARIOTYPE where LIBELLE="" &  
%SCENARIO_TYPE_NAME & """;
```

```
COLUMN DEFINITION TYPE = "STRING";
```

```
%LET DEFINTIION = DEFINITION;
```

```
RUN;
```

```
PROC EFRONTBLOB BLOB=%DEFINTIION PATH=PERIODDEFINITON  
FORMAT="COLLECTION" OUT=WORK.TEST;
```

```
RUN;
```

```
PROC PRINT DATA=WORK.TEST;
```

RUN:

### 5.5.1.4. Result

LP (P) Portfolio Company							
	COLUMN	LIBELLE	NAME	ALIGNMENT	RO	RO_R	FORMAT
1	STARTPERIOD1	Market Value Date	Market Value Date	1	False	False	,
2	REVENUE1	Partnership Market	Partnership Market	1	False	False	,##0;(#,##0)
3	COST1	Partnership Total	Partnership Total	1	False	False	,##0;(#,##0)
4	OTHERINCOME1	Partnership	Partnership	1	False	False	,##0;(#,##0)
5	OVERHEADS1	Partnership	Partnership	1	False	False	,##0;(#,##0)
6	EBIT1	Partnership Return	Partnership Return	1	False	False	,##0;(#,##0)
7	EBITDA1	Partnership	Partnership	1	False	False	,##0;(#,##0)
8	DEPRECIATION1	Partnership	Partnership	1	False	False	,##0;(#,##0)
9	ABNORMALITEMS1	Exposed Market	Exposed Market	1	False	False	,##0;(#,##0)
10	PROFITBT1	Exposed Total	Exposed Total	1	False	False	,##0;(#,##0)
11	TAX1	Exposed Remaining	Exposed Remaining	1	False	False	,##0;(#,##0)
12	PROFIT1	Exposed Proceeds	Exposed Proceeds	1	False	False	,##0;(#,##0)
13	ASSETS11	Exposed Return of	Exposed Return of	1	False	False	,##0;(#,##0)
14	ASSETS21	Exposed Unrealized	Exposed Unrealized	1	False	False	,##0;(#,##0)
15	STOCK1	Exposed Realized	Exposed Realized	1	False	False	,##0;(#,##0)
16	SHORTTERMDEBT1	Partnership Market	Partnership Market	1	False	False	,##0;(#,##0)
17	LONGTERMDEBT1	Partnership Total	Partnership Total	1	False	False	,##0;(#,##0)
18	OPERATIONALDEBT	Partnership	Partnership	1	False	False	,##0;(#,##0)
19	CASHFLOW1	Partnership	Partnership	1	False	False	,##0;(#,##0)
20	INVESTMENTS1	Partnership Return	Partnership Return	1	False	False	,##0;(#,##0)
21	INVESTMENTS11	Partnership	Partnership	1	False	False	,##0;(#,##0)
22	RESEARCHANDDEV	Partnership	Partnership	1	False	False	,##0;(#,##0)
23	CASHAVAILABLE1	Exposed Market	Exposed Market	1	False	False	,##0;(#,##0)
24	ASSETS01	Exposed Total	Exposed Total	1	False	False	,##0;(#,##0)
25	CREDITORS11	Exposed Remaining	Exposed Remaining	1	False	False	,##0;(#,##0)

### 5.5.2. PROC EFRONTBLOB - DATA - CLASS - PREFIX - ID

#### 5.5.2.1. Goal

Get all the blob fields of the funds, and import them into an **eFront Report** data table. The blob fields to be exported are regrouped in a table of blob fields.

#### 5.5.2.2. Features being illustrated

- PROC EFRONTBLOB step

- %LET statement

### 5.5.2.3. Program

```
%LET SQL_QUERY = "SELECT DISTINCT A.IQID, A.FUND,
CONVERT(VARCHAR(8000),A1.FUND_INFO)FROM VCFUND A LEFT JOIN
VCFUNDOOTHER A1 ON A1.IQID = A.FUNDOTHER WHERE ??FILTER";

DATA WORK.FUND;

SQL %_QUERY;

COLUMN FUND_ID LABEL="" TYPE=STRING;

COLUMN FUNDNAME LABEL="Fund" TYPE=STRING;

COLUMN FUND_INFO TYPE=STRING;

RUN;

PROC EFRONTBLOB DATA=WORK.FUND ID=FUND_INFO CLASS="FUND"
PATH="MYINFO" PREFIX="BLOB_FUND_" OUT=WORK.WORK.BLOB_1;

VAR FUND_ID.

RUN;

DATA WORK.FUND;

MERGE WORK.FUND(IN=T1) WORK.BLOB_1;

BY FUND_ID;

_OUTPUT_ = T1;

RUN;
```

### 5.5.2.4. Comment

In the example above, the table **FUND\_INFO** contains the column **BLOB**, which refers to all the blob fields associated with the funds (the additional fields of the VCFUND table). This table is called by the **SQL\_QUERY** macro variable.

When configuring the application, and creating additional fields of type BLOB, you define a **CLASS**es, which allows you to regroup **BLOB** fields under classes. The CLASS option in the EFRONTBLOB step refers to this class.

The PREFIX option refers to the prefix to be used to signal the BLOB fields in the columns of the output table.

## 5.6. Examples - PROC EFRONTDASHBOARD step

The following examples illustrate the PROC EFRONTFOLDER step and its subordinate statements:

- [PROC EFRONTDASHBOARD - PAGES](#)

### 5.6.1. PROC EFRONTDASHBOARD - PAGES

#### 5.6.1.1. Goal

Export a dashboard to a PDF file.

#### 5.6.1.2. Features being illustrated

- [PROC EFRONTDASHBOARD step](#)
- [PAGES option](#)

#### 5.6.1.3. Program

```
PROC EFRONTDASHBOARDDDASHBOARD="Dashboard_folder\Dashboard_name"  
PAGES="1-3,5";
```

RUN;

#### 5.6.1.4. Result

The program outputs a **Dashboard\_name.pdf** file containing pages 1,2,3 and 5 of the source dashboard.

#### 5.6.1.5. Comment

- If the PAGES option is not used, the program exports all pages of the source dashboard.

### 5.6.2. PROC EFRONTDASHBOARD - dashboard parameters

#### 5.6.2.1. Goal

Use dashboard parameters to export data for a specific fund.

### 5.6.2.2. Features being illustrated

- PROC EFRONTDASHBOARD

### 5.6.2.3. Introduction

This example uses a dashboard fed by the **LIBRARY\_ALLASSETS\_PERFORMANCE** FrontScript library, which has the following parameters:

- PORTFOLIO
- ASOFDATE
- DASHBOARD\_CURRENCY

### 5.6.2.4. Program

```
%LETiqid='6D4CBCAEC58240D4BA0FF2AA7C0D9A48'; // the fund's IQID  
%LETrefdate='10/12/2019';  
%LETcurrency='USD';  
  
PROC  
EFRONTDASHBOARDDASHBOARD="Dashboard_folder\Dashboard"(PORTFOLIO  
=%iqid ASOFDATE=%refdate DASHBOARD_CURRENCY=%currency);  
  
RUN;
```

### 5.6.2.5. Result

A PDF with data for eFront World PE fund, as of December 10th 2019, with additional columns for figures in the dashboard currency.

### 5.6.2.6. Comment

- Note that you must use the IQID of the fund in order for the program to retrieve the fund's data.

## 5.7. Examples - PROC EFRONTFOLDER step

The following examples illustrate the PROC EFRONTFOLDER step:

- [PROC EFRONTFOLDER - FOLDER - ID - PROPERTIES](#)
- [PROC EFRONTFOLDER - Get portfolio positions](#)

### 5.7.1. PROC EFRONTFOLDER - FOLDER - ID - PROPERTIES

#### 5.7.1.1. Goal

Get...

#### 5.7.1.2. Features being illustrated

- [PROC EFRONTFOLDER step](#)

#### 5.7.1.3. Program

//...

```
%DEFINE REPORT_DATE;
```

```
%PARAM DATE LABEL="End Date{F}Date D'Arrêté" TYPE=DATE DEFAULT=TODAY  
NOTNULL;
```

```
%LET PROPERTIES = "FORCERECALC=1;XX_DATE=" & %DATE;
```

```
PROC EFRONTFOLDER FOLDER="F_AJXPEINSTRUMENTPOS"  
ID="6D4B5DF1BB6647619A6E5FA2EBAFC197" PROPERTIES=%PROPERTIES;
```

```
RUN;
```

#### 5.7.1.4. Comment

- ....

## 5.7.2. PROC EFRONTFOLDER - Get portfolio positions

### 5.7.2.1. Goal

Get the **Portfolio's Position** corresponding to the **Actual figures up to** property of the financial scenario.

**Fund Scenario - Identity**

Recalculate	Reference date: 22/03/2010	Recalculate	Investor: Private Equity Capital Germany																																																																																	
Actual figures up to:	22/03/2010	Explicit % ownership:	* Apply to Portfolio's Position																																																																																	
Calculation Mode:		Target Currency:	* Default is Investor's Currency																																																																																	
<b>Position</b> <table border="1"> <tr> <th>Fund Currency</th> <th>Investor Currency</th> </tr> <tr> <td>USD</td> <td>EUR</td> </tr> <tr> <td>Total commitment</td> <td>14 000 000,00</td> </tr> <tr> <td>Remaining commitment</td> <td>9 476 268,00</td> </tr> <tr> <td>% Called</td> <td>35,68%</td> </tr> <tr> <td>Current drawdowns</td> <td>4 895 765,00</td> </tr> <tr> <td>Net Invested</td> <td>4 108 332,00</td> </tr> <tr> <td>NAV</td> <td>3 664 191,00</td> </tr> <tr> <td></td> <td>2 553 889,83</td> </tr> </table>		Fund Currency	Investor Currency	USD	EUR	Total commitment	14 000 000,00	Remaining commitment	9 476 268,00	% Called	35,68%	Current drawdowns	4 895 765,00	Net Invested	4 108 332,00	NAV	3 664 191,00		2 553 889,83	<b>Closing Date:</b> <table border="1"> <tr> <td>Closing date: 31/03/2007</td> <td>% Ownership: 100,00%</td> </tr> <tr> <td colspan="2"> <b>Portfolio's Valuation &amp; Exit Assumptions</b> </td> </tr> <tr> <td colspan="2">Force Current Portfolio Multiple: [ ]</td> </tr> <tr> <td colspan="2">Model: [ ] Overridden by: [ ]</td> </tr> <tr> <td colspan="2">Expected Holding period (in years): 4</td> </tr> <tr> <td colspan="2">Expected Multiple: 2,0736</td> </tr> <tr> <td colspan="2">First Revaluation period (in years): 2,5</td> </tr> <tr> <td colspan="2">Under Valuation discount: 30,00%</td> </tr> </table>		Closing date: 31/03/2007	% Ownership: 100,00%	<b>Portfolio's Valuation &amp; Exit Assumptions</b>		Force Current Portfolio Multiple: [ ]		Model: [ ] Overridden by: [ ]		Expected Holding period (in years): 4		Expected Multiple: 2,0736		First Revaluation period (in years): 2,5		Under Valuation discount: 30,00%																																																
Fund Currency	Investor Currency																																																																																			
USD	EUR																																																																																			
Total commitment	14 000 000,00																																																																																			
Remaining commitment	9 476 268,00																																																																																			
% Called	35,68%																																																																																			
Current drawdowns	4 895 765,00																																																																																			
Net Invested	4 108 332,00																																																																																			
NAV	3 664 191,00																																																																																			
	2 553 889,83																																																																																			
Closing date: 31/03/2007	% Ownership: 100,00%																																																																																			
<b>Portfolio's Valuation &amp; Exit Assumptions</b>																																																																																				
Force Current Portfolio Multiple: [ ]																																																																																				
Model: [ ] Overridden by: [ ]																																																																																				
Expected Holding period (in years): 4																																																																																				
Expected Multiple: 2,0736																																																																																				
First Revaluation period (in years): 2,5																																																																																				
Under Valuation discount: 30,00%																																																																																				
<b>Portfolio's Position (Fund Currency)</b> <table border="1"> <thead> <tr> <th>Company</th> <th>First disburse.</th> <th>Exit date</th> <th>Total cost</th> <th>Current cost</th> <th>Current proce...</th> <th>Valuation</th> <th>Calc'd Exit D...</th> <th>Calc'd Co...</th> </tr> </thead> <tbody> <tr> <td>BankUnited FSB</td> <td>09/06/2009</td> <td></td> <td>101 830,00</td> <td>101 830,00</td> <td>0,00</td> <td>112 013,00</td> <td>31/12/2013</td> <td>101 830,00</td> </tr> <tr> <td>Bloo... Allen</td> <td>23/07/2008</td> <td></td> <td>913 115,00</td> <td>389 937,00</td> <td>523 178,00</td> <td>846 494,00</td> <td>30/09/2010</td> <td>913 115,00</td> </tr> <tr> <td>CARRIED</td> <td></td> <td></td> <td>0,00</td> <td>0,00</td> <td>0,00</td> <td>0,00</td> <td>21/03/2010</td> <td>0,00</td> </tr> <tr> <td>CASH</td> <td>06/07/2007</td> <td></td> <td>91 786,00</td> <td>87 848,00</td> <td>3 938,00</td> <td>0,00</td> <td>06/07/2011</td> <td>91 786,00</td> </tr> <tr> <td>CPV Traded Debt Instruments</td> <td>01/07/2009</td> <td></td> <td>6 472,00</td> <td>6 472,00</td> <td>0,00</td> <td>20 712,00</td> <td>30/09/2010</td> <td>6 472,00</td> </tr> <tr> <td>CVC Brasil</td> <td>01/01/2010</td> <td></td> <td>204 295,00</td> <td>204 295,00</td> <td>0,00</td> <td>193 201,00</td> <td>01/01/2014</td> <td>204 295,00</td> </tr> <tr> <td>HD Supply</td> <td>27/09/2007</td> <td></td> <td>865 751,00</td> <td>865 751,00</td> <td>0,00</td> <td>174 537,00</td> <td>31/12/2012</td> <td>865 751,00</td> </tr> <tr> <td>Manor Care</td> <td>06/11/2007</td> <td></td> <td>1 150 284,00</td> <td>1 150 284,00</td> <td>0,00</td> <td>1 149 743,00</td> <td>31/12/2012</td> <td>1 150 284,00</td> </tr> </tbody> </table>				Company	First disburse.	Exit date	Total cost	Current cost	Current proce...	Valuation	Calc'd Exit D...	Calc'd Co...	BankUnited FSB	09/06/2009		101 830,00	101 830,00	0,00	112 013,00	31/12/2013	101 830,00	Bloo... Allen	23/07/2008		913 115,00	389 937,00	523 178,00	846 494,00	30/09/2010	913 115,00	CARRIED			0,00	0,00	0,00	0,00	21/03/2010	0,00	CASH	06/07/2007		91 786,00	87 848,00	3 938,00	0,00	06/07/2011	91 786,00	CPV Traded Debt Instruments	01/07/2009		6 472,00	6 472,00	0,00	20 712,00	30/09/2010	6 472,00	CVC Brasil	01/01/2010		204 295,00	204 295,00	0,00	193 201,00	01/01/2014	204 295,00	HD Supply	27/09/2007		865 751,00	865 751,00	0,00	174 537,00	31/12/2012	865 751,00	Manor Care	06/11/2007		1 150 284,00	1 150 284,00	0,00	1 149 743,00	31/12/2012	1 150 284,00
Company	First disburse.	Exit date	Total cost	Current cost	Current proce...	Valuation	Calc'd Exit D...	Calc'd Co...																																																																												
BankUnited FSB	09/06/2009		101 830,00	101 830,00	0,00	112 013,00	31/12/2013	101 830,00																																																																												
Bloo... Allen	23/07/2008		913 115,00	389 937,00	523 178,00	846 494,00	30/09/2010	913 115,00																																																																												
CARRIED			0,00	0,00	0,00	0,00	21/03/2010	0,00																																																																												
CASH	06/07/2007		91 786,00	87 848,00	3 938,00	0,00	06/07/2011	91 786,00																																																																												
CPV Traded Debt Instruments	01/07/2009		6 472,00	6 472,00	0,00	20 712,00	30/09/2010	6 472,00																																																																												
CVC Brasil	01/01/2010		204 295,00	204 295,00	0,00	193 201,00	01/01/2014	204 295,00																																																																												
HD Supply	27/09/2007		865 751,00	865 751,00	0,00	174 537,00	31/12/2012	865 751,00																																																																												
Manor Care	06/11/2007		1 150 284,00	1 150 284,00	0,00	1 149 743,00	31/12/2012	1 150 284,00																																																																												

### 5.7.2.2. Features being illustrated

- PROC EFRONTFOLDER step

### 5.7.2.3. Program

//...

```
%INCLUDE P_DATE;  
  
%LET IQID_Scenario = "12A08834C05748E49E0DC64B68F4F144";  
  
%LETPROPERTIES = "Cutoffdate=" & %Date;  
  
PROC EFRONTFOLDER FOLDER="F_AJXPECFSCENARIO" ID=%IQID_Scenario  
PROPERTIES=%Properties;  
  
RUN;
```

### 5.7.2.4. Comment

- ....

## 5.8. Examples - PROC EFRONTIMPORT step

The following examples illustrate the PROC EFRONTIMPORT step, and subordinate statements:

- [DATA - INFILE - PROC EFRONTIMPORT](#)

### 5.8.1. DATA - INFILE - PROC EFRONTIMPORT

#### 5.8.1.1. Goal

Import quotations of listed instruments from an EXCEL spreadsheet.

#### 5.8.1.2. Features being illustrated

- [PROC EFRONTIMPORT step](#)
- [DATA step](#)
- [INFILE statement](#)

#### 5.8.1.3. Program

//Import quotations data from an EXCEL spreadsheet, and create the data table T

LIBNAME USER ":";

%LET DIRECTORY = "c:\bloomberg";

%LET FILE\_PATH = %DIRECTORY & "\bloomberg.xls";

TRACE %FILE\_PATH;

DATA WORK.T;

INFILE "C:\TMP\BLOOMBERG.XLS" AREAS="C7:C999;J2;A7:A999;J7:J999";

COLUMN COMPANY;

COLUMN DATE TYPE=DATE;

COLUMN ISIN;

```
COLUMN PRICE TYPE=FLOAT;  
  
//import only if no cell is empty  
  
IF (COMPANY IS NOT NULL) AND (ISIN IS NOT NULL) AND (PRICE IS NOT NULL)  
THEN  
  
OUTPUT;  
  
END;  
  
RUN;  
  
//Import the data from the eFront Report data table applying the import mask  
  
PROC EFRONTIMPORT DATA=WORK.T NAME="##BLOOMBERG##" DETAILS;  
  
RUN;  
  
PROC PRINT DATA = WORK.T;  
  
RUN;  
  
PROC PRINT;  
  
PUT %OUTPUTLOG;  
  
RUN;
```

#### 5.8.1.4. Comment

- C7:C999 contains the company name that is imported into the **COMPANY** column.
- J2 contains the date that is imported into the **DATE** column.
- A7:A999 contains the ISIN code that is imported into the **ISIN** column.
- J7:J999 contains the price code that is imported into the **PRICE** column.

## 5.9. Examples - PROC EFRONTIMPORT\_AJX step

The following examples illustrate the PROC EFRONTIMPORT step, and subordinate statements:

- [DATA - INFILE - PROC EFRONTIMPORT](#)

### 5.9.1. PROC EFRONTIMPORT\_AJX - %OUTPUTLOG

#### 5.9.1.1. Goal

Import an **eFront Report** table in a specific sheet from the file of a given FIA import registered in the system.

#### 5.9.1.2. Features being illustrated

- [PROC EFRONTIMPORT\\_AJX](#) step

#### 5.9.1.3. Program

//

```
PROC EFRONTIMPORT_AJX DATA=WORK.T_MYTABLE NAME="FIA_Import"  
SHEETNAME="Sheet1" DETAILS;
```

RUN;

PROC PRINT;

PUT %OUTPUTLOG;

RUN;

#### 5.9.1.4. Result

Source **eFront Report** data table such as the table below:

FUND_NAME	INVESTOR_NAME	MGT_FEES	OPE_DATE	OPERATION	INDEX	TYPE
INFRA II	01 European	248099.668600838	05/12/2011	Call 19	1	Call
INFRA II	02 APG	708856.198002395	05/12/2011	Call 19	1	Call

is imported into the sheet of an **FIA standard import** as below:

	A	B	C	D	E	F
1	ef\$class	ef\$subclass	ef\$key	ef\$col	ef\$col	ef\$key
2	FundOperation	WithShares	Fund.Fund	Investors (BP).Subscriber.Fund investment	Investors (BP).Share Mgmt Fees In Commitment	Date
3						

### 5.9.1.5. Comment

- The columns surrounded by red don't need to be included in the source eFront Report table, but all the columns at the left of the ones entitled **ef\$class** and **ef\$sublass** in the FIA import, need to be present in the source **eFront Report** data table.

## 5.10. Examples - PROC EFRONTMENUS step

The following examples illustrate the PROC EFRONTMENUS step, and subordinate statements:

- [PROC EFRONTMENUS - GETUSERPROFILE - GETUSERID](#)

### 5.10.1. PROC EFRONTMENUS - GETUSERPROFILE - GETUSERID

#### 5.10.1.1. Goal

Display for the current eFront Invest user all the accessible menus and sub-menus.



To obtain the following

#### 5.10.1.2. Features being illustrated

- [PROC EFRONTMENUS step](#)
- [Functions - special](#)

#### 5.10.1.3. Program

PROC EFRONTMENUS

```
PROFILE=GETUSERPROFILE(GETUSERID())
```

```
OUT=WORK.TMP;
```

```
RUN;
```

```
PROC PRINT DATA=WORK.TMP;
```

```
RUN;
```

## 5.10.1.4. Result

### PROC EFRONTMENUS

Execute					
	PROFILE_ID	MENU_INDEX	MENU_CAPTION	SUBMENU_INDEX	SUBMENU_CAPTION
1	3B1A634D21CC...	1 Open		1 Investors	
2	3B1A634D21CC...	1 Open		2 Fundraising Pipel...	
3	3B1A634D21CC...	1 Open		3 {separator}	
4	3B1A634D21CC...	1 Open		4 Deals	
5	3B1A634D21CC...	1 Open		5 Investment Struct...	
6	3B1A634D21CC...	1 Open		6 {separator}	
7	3B1A634D21CC...	1 Open		7 Funds	
8	3B1A634D21CC...	1 Open		8 Companies	
9	3B1A634D21CC...	1 Open		9 Properties	
10	3B1A634D21CC...	1 Open		10 Hedge Instruments	
11	3B1A634D21CC...	1 Open		11 Portfolios	
12	3B1A634D21CC...	1 Open		12 {separator}	
13	3B1A634D21CC...	1 Open		13 Fund Operations	
14	3B1A634D21CC...	1 Open		14 Transactions	
15	3B1A634D21CC...	1 Open		15 Mass update inte...	
16	3B1A634D21CC...	1 Open		16 Mass Undraft	
17	3B1A634D21CC...	1 Open		17 {separator}	
18	3B1A634D21CC...	1 Open		18 Files	
19	3B1A634D21CC...	2 CRM		1 Contacts	
20	3B1A634D21CC...	2 CRM		2 Appointments	

## 5.11. Examples - PROC EFRONTTABLES step

The following examples illustrate the PROC EFRONTTABLES step, and subordinate statements:

- [PROC EFRONTTABLES - TABLES - INDEXES - RELATIONS](#)

### 5.11.1. PROC EFRONTTABLES - TABLES - INDEXES - RELATIONS

#### 5.11.1.1. Goal

Generate data tables that describe the DB data model, including system tables.

#### 5.11.1.2. Features being illustrated

- [PROC EFRONTTABLES step](#)

#### 5.11.1.3. Program

//Generate data tables that describe the DB data model

```
PROC EFRONTTABLES  
TABLES=T_TABLESCOLUMNS=T_COLUMNSINDEXES=T_INDEXESRELATIONS=T_  
RELATIONSINCLUDESYSTABLES;
```

RUN;

```
PROC PRINTDATA=T.TABLES;
```

RUN;

## 5.11.14. Result

	Name	Caption	UserTypes	Lookup	Maintable	UserLookup	Package
1	ADMACCESS	ADMACCESS	False	False	True	False	PCKFRONTADMIN
2	ADMACCESSROLE		False	False	False	False	PCKFRONTADMIN
3	ADMACCOUNT	ADM/Account	True	False	True	False	PCKFRONTADMIN
4	ADMACL	ADM/ACL	False	False	True	False	PCKFRONTADMIN
5	ADMACTION	Action	False	False	True	False	PCKFRONTADMIN
6	ADMACTTYPE	Action types	False	True	False	False	PCKFRONTADMIN
7	ADMALERTPRIORI	ADMALERTPRIORI	False	True	False	True	PCKFRONTADMIN
8	ADMASYNCH	ADMASYNCH	False	False	False	False	PCKFRONTADMIN
9	ADMAUDIT	Audit	False	False	True	False	PCKFRONTADMIN
10	ADMAUDITCOLS	Audit Columns	True	False	True	False	PCKFRONTADMIN
11	ADMAUDITEVENT	ADMAUDITEVENT	False	False	False	False	PCKFRONTADMIN
12	ADMAUDITTYPE	ADM/Audit Types	False	True	False	False	PCKFRONTADMIN
13	ADMAUTHTOKEN	admauthtoken	False	False	False	False	PCKFRONTADMIN
14	ADMBILLINGENTR	Billing entry	False	False	True	False	PCKFRONTADMIN
15	ADMCATEGORY	Reports category	False	False	True	False	PCKFRONTADMIN
16	ADMCONTENT	Content	False	False	True	False	PCKFRONTADMIN
17	ADMDEFWFSTATUS	ADM/System	False	True	False	True	PCKFRONTADMIN
18	ADMDEFWRKFLW	ADM/System	True	False	True	False	PCKFRONTADMIN
19	admdefwrkflwobject	Workflow/Object	False	False	False	False	PCKFRONTADMIN
20	ADMDELAYEDMAIL	ADMDELAYEDMAIL	False	True	False	False	PCKFRONTADMIN
21	ADMDELAYEDMAIL	ADM/Delayed Mail	False	True	False	True	PCKFRONTADMIN
22	ADMDEPARTDEPA	ADM/Department/D	False	False	True	False	PCKFRONTADMIN
23	ADMDEPARTMENTORGANISATIONAL		True	False	True	False	PCKFRONTADMIN
24	ADMDEPARTMENT		True	False	True	False	PCKFRONTADMIN
25	ADMDESKTOP	ADM/Desktop	False	False	True	False	PCKFRONTADMIN
26	ADMDESKTOPALE	ADMDESKTOPALE	True	False	True	False	PCKFRONTADMIN
27	ADMDISKPATHS	ADM/Disk Paths	False	False	False	False	PCKFRONTADMIN
28	ADMDOPTLINKTYPE	ADM/Departments'	False	True	False	True	PCKFRONTADMIN
29	ADMDOPTTYPE	ADM/Departments	False	True	False	True	PCKFRONTADMIN
30	ADMMDWHCONTEXT	ADMMDWHCONTEXT	False	False	False	False	PCKFRONTADMIN
31	ADMODYNLINK	ADM/Dynamic links	False	False	False	False	PCKFRONTADMIN
32	ADMODYNROLE	ADM/Dynamic	False	False	False	False	PCKFRONTADMIN
33	ADMEVENTTRACK	ADMEVENTTRACK	False	False	True	False	PCKFRONTADMIN
34	ADMEXTENDEDPRE	ADM/Extended	False	False	True	False	PCKFRONTADMIN
35	ADMFILES	ADM/Files	True	False	True	False	PCKFRONTADMIN
36	ADMFILTERCURRE	ADM/Currency	False	True	False	True	PCKFRONTADMIN
37	ADMFOLDERRATE	ADM/Folder rate	False	False	False	False	PCKFRONTADMIN
38	ADMFORMULA	Formula	False	False	True	False	PCKFRONTADMIN
39	ADMFUNCTION	ADM/Functions	False	True	False	True	PCKFRONTADMIN
40	ADMGLOBALS	ADM/Globals	False	False	False	False	PCKFRONTADMIN

## 5.12. Examples - PROC EXPORT step

The following examples illustrate the PROC EXPORT step:

- [PROC EXPORT \(1\)](#)
- [PROC EXPORT FORMAT\(\)](#)
- [PROC EXPORT - %LET - PROC FORMAT](#)

### 5.12.1. PROC EXPORT (1)

#### 5.12.1.1. Goal

Export a data table to a .CSV file on the local server disk.

#### 5.12.1.2. Features being illustrated

- [PROC EXPORT step](#)

#### 5.12.1.3. Program

//Create a macro-variable for the file name

```
%LET BULK_EXPORT_FILE1 =  
"C:\efront\website\download\TABLE_ADRESSES.CSV";
```

//Export the data table

```
PROC EXPORT DATA=WORK.T_ADRESSES DROP NOERROR  
FILE=%BULK_EXPORT_FILE1 NOHEADER SEMICOLUMN;
```

RUN;

#### 5.12.1.4. Result

The application generates a .CSV file from the table T\_ADRESSES. Semicolons are used as separators. The export file doesn't integrate the column header from the source data table.

## 5.12.2. PROC EXPORT - FORMAT()

### 5.12.2.1. Goal

Create a .TXT file on the server disk. The file name of the file to create is based on the report run dates in the format of MMDDYYYY (e.g. '01072009' instead of '172009').

### 5.12.2.2. Features being illustrated

- [PROC EXPORT step](#)
- Functions - dates

### 5.12.2.3. Program

//Create a macro-variable for the file name

```
%LET FILENAME_TAB = "E:\efront\PACE\Automatic\Transactions\efr_trans_" &
FORMAT(%STARTDATE, "ddmmyyyy") & "_" & FORMAT(%ENDDATE, "ddmmyyyy") &
".TXT";
```

//Export the data table

```
PROC EXPORT DATA=WORK.T_TRANSACTIONS_EXPORT DROP NOERROR
FILE=%FILENAME_TAB TAB;
```

RUN;

### 5.12.2.4. Result

The application generates a .TXT file named in the following way:

E:\efront\PACE\Automatic\Transactions\efr\_trans\_01072009\_01072009.TXT

## 5.12.3. PROC EXPORT - %LET - PROC FORMAT

### 5.12.3.1. Goal

Export the eFront Report output file to a local server, which is not the server that hosts the application.

### 5.12.3.2. Features being illustrated

- PROC EXPORT step
- PROC FORMAT step
- %LET statement

### 5.12.3.3. Program

// Create common header used for a group of reports. Within the header we define the macro-variable EXPORT\_PATH whose value is set to the path to the non-local server.

```
LIBNAME EXTRACT_VIEWS "\eFront (Shared)\Extracts\VIEWS";  
LIBNAME EXTRACT_DATES "\eFront (Shared)\EXTRACTS\LAST_EXTRACT_DATE";  
PROC FORMAT;  
PICTURE Datetime Low-High = "yyy-MM-dd-HH:mm:ss";  
RUN;  
%LET EXPORT_FORMAT=".csv";  
%LET EXPORT_PATH="\  
\LON1MIS01\SVG\EFRONT_SOURCE_FILES\CAPITAL_FUNDS\";  
%LET LAST_EXTRACT_TABLE = "T_LAST_EXTRACT_DATE";  
//Define specific report; In the following, we only extract the lines that are useful to  
illustrate the example  
//. Export to csv file  
%IF TABLENOBS("WORK.FX_RATES") > 1  
%THEN  
//Create a macro-variable for access path to the CSV table  
%LET INSTANCE="SVGC_";
```

```
%LET FILE_NAME="EXCHANGERATE_";  
%LET M_T="TRANS_";  
%LET DATE_STAMP = FORMAT(%CURRENT_DATE,"yyyy-MM-dd-HHmmss");  
//Export the temporary data table and create an CSV file on the server  
PROC EXPORT DATA=WORK.FX_RATES COMMA DROP NOHEADER LABEL  
FILE=%EXPORT_PATH &%INSTANCE & %FILE_NAME & %M_T & %DATE_STAMP  
& %EXPORT_FORMAT;  
RUN;  
TRACE "FX Rates Extracted";  
%END;  
DATA EXTRACT_DATES.@LAST_EXTRACT_TABLE;  
SET EXTRACT_DATES.@LAST_EXTRACT_TABLE;  
@EXTRACT_NAME = %CURRENT_DATE;  
RUN;  
%UNDEFINE CURRENT_DATE;  
%UNDEFINE FILTER_LAST_EXTRACT;  
%UNDEFINE LAST_EXTRACT_DATE;  
//..
```

## 5.12.3.4. Result

LP (Pl) Portfolio Company							
COLUMN		LIBELLE	NAME	ALIGNMENT	RO	RO_R	FORMAT
1	STARTPERIOD1	Market Value Date	Market Value Date	1	False	False	
2	REVENUE1	Partnership Market	Partnership Market	1	False	False	,##0;(#,##0)
3	COST1	Partnership Total	Partnership Total	1	False	False	,##0;(#,##0)
4	OTHERINCOME1	Partnership	Partnership	1	False	False	,##0;(#,##0)
5	OVERHEADS1	Partnership	Partnership	1	False	False	,##0;(#,##0)
6	EBIT1	Partnership Return	Partnership Return	1	False	False	,##0;(#,##0)
7	EBITDA1	Partnership	Partnership	1	False	False	,##0;(#,##0)
8	DEPRECIATION1	Partnership	Partnership	1	False	False	,##0;(#,##0)
9	ABNORMALITEMS1	Exposed Market	Exposed Market	1	False	False	,##0;(#,##0)
10	PROFITBT1	Exposed Total	Exposed Total	1	False	False	,##0;(#,##0)
11	TAX1	Exposed Remaining	Exposed Remaining	1	False	False	,##0;(#,##0)
12	PROFIT1	Exposed Proceeds	Exposed Proceeds	1	False	False	,##0;(#,##0)
13	ASSETS11	Exposed Return of	Exposed Return of	1	False	False	,##0;(#,##0)
14	ASSETS21	Exposed Unrealized	Exposed Unrealized	1	False	False	,##0;(#,##0)
15	STOCK1	Exposed Realized	Exposed Realized	1	False	False	,##0;(#,##0)
16	SHORTTERMDEBT1	Partnership Market	Partnership Market	1	False	False	,##0;(#,##0)
17	LONGTERMDEBT1	Partnership Total	Partnership Total	1	False	False	,##0;(#,##0)
18	OPERATIONALDEBT	Partnership	Partnership	1	False	False	,##0;(#,##0)
19	CASHFLOW1	Partnership	Partnership	1	False	False	,##0;(#,##0)
20	INVESTMENTS1	Partnership Return	Partnership Return	1	False	False	,##0;(#,##0)
21	INVESTMENTS11	Partnership	Partnership	1	False	False	,##0;(#,##0)
22	RESEARCHANDEV	Partnership	Partnership	1	False	False	,##0;(#,##0)
23	CASHAVAILABLE1	Exposed Market	Exposed Market	1	False	False	,##0;(#,##0)
24	ASSETS01	Exposed Total	Exposed Total	1	False	False	,##0;(#,##0)
25	CREDITORS11	Exposed Remaining	Exposed Remaining	1	False	False	,##0;(#,##0)

## 5.13. Examples - PROC EXPORTCHART step

The following examples illustrate the PROC EXPORTCHART step:

- [PROC EXPORTCHART 1 - Generate graphs and integrate them into a WORD doc](#)
- [PROC EXPORTCHART 2 - Generate Graphs and integrate them into a WORD Doc](#)

 To well understand how PROC EXPORTCHART works, we recommend you study the above examples in the suggested order!

### 5.13.1. PROC EXPORTCHART 1 - Generate graphs and integrate them into WORD docs

#### 5.13.1.1. Goal

Generate graphs in an EXCEL file, extract them as JPEG files and integrate them into a WORD document.

 The example below gives you a very detailed vision on how to use the step PROC EXPORTCHART!

#### 5.13.1.2. Features being illustrated

- [PROC EXPORTCHART step](#)
- [PROC OFFICE step](#)
- [DATA step](#)

#### 5.13.1.3. Algorithm used to implement example in eFront Script

- Create EXCEL report with charts.  
Use PROC OFFICE step.
- Extract the charts from the EXCEL report.  
Use PROC EXPORTCHART step.
- Create a data table which refers to the data to be used in the WORD report.  
Use the Data step.

 When building a WORD report using the PROC OFFICE step, which will be done to produce the final WORD report, you need to supply to the PROC OFFICE step not

only the template to be used, but also a data table which specifies the data to be used in the WORD report. This table needs to be built prior to the construction of the WORD report. This table needs to contain the names of the image files to be used in the WORD report, and eventually, some WORD template syntax.

 Be aware that you need to create for each image in an EXCEL page, a specific chart column!

- Create WORD report based on a WORD template + the data table which specifies the charts to be integrated.  
Use the PROC OFFICE step.

```
PROC OFFICE FILE="R_FX_RATES_TASK3.XLSX";
SET WORK.CCY WORK.T_RATES WORK.T_RATES_LIST WORK.T_RATES_DATES WORK.T_RATES_S WORK.T_RATES_MEANS;
TEMPLATE "&R_TASK3_0.xlsx";
RUN;

PROC EXPORTCHART FILE="R_FX_RATES_TASK3.XLSX"; RUN; 1

DATA CHART_LIST;
SET TABLES.CCY;
COLUMN DESC;
COLUMN START_DATE TYPE = DATE;
COLUMN END_DATE TYPE = DATE;
COLUMN CHART;
COLUMN CHART2;
DESC = %SOURCE_CCY% to "%CURRENCY1";
START_DATE = %START_DATE;
END_DATE = %END_DATE;
CHART = "'&DESC&'-Chart 1.jpeg";
CHART2 = "'&DESC&'-Chart 2.jpeg";OUTPUT;
RUN;

PROC OFFICE FILE="D:\test\FX Charts.DOC";
SET CHART_LIST;
TEMPLATE "D:\add task\FX Charts copy2.DOC";
RUN; 2
```

3
4

 In the example above, step 1 is preceded by the instructions being necessary to built the content of the EXCEL file. In the example description, we only focus on the program excerpts which are relevant to illustrate the usage of the procedure **PROC EXPORTCHART**.

#### 5.13.1.4. Program

...

//Create EXCEL report with charts directly from within the program

```
PROC OFFICE FILE="R_FX_RATES_TASK3.XLSX";
SET T_RATES T_RATES_S;
TEMPLATE "&FXRates_KB.xlsx";
```

RUN;

```
//Export charts from previously created file. The procedure creates images in format  
"Sheet name-Chart name.jpeg"
```

```
PROC EXPORTCHART FILE="R_FX_RATES_TASK3.XLSX";
```

RUN;

```
//Create data table with image names and some other data for word template
```

```
DATA CHART_LIST;
```

```
SET CCY;
```

```
COLUMN DESC;
```

```
COLUMN START_DATE TYPE = DATE;
```

```
COLUMN END_DATE TYPE = DATE;
```

```
COLUMN CHART;
```

```
COLUMN CHART2;
```

```
//DESC column with cross pair name
```

```
DESC = %SOURCE_CCY&" to "&CURRENCY1;
```

```
//input parameters
```

```
START_DATE = %START_DATE;
```

```
END_DATE = %END_DATE;
```

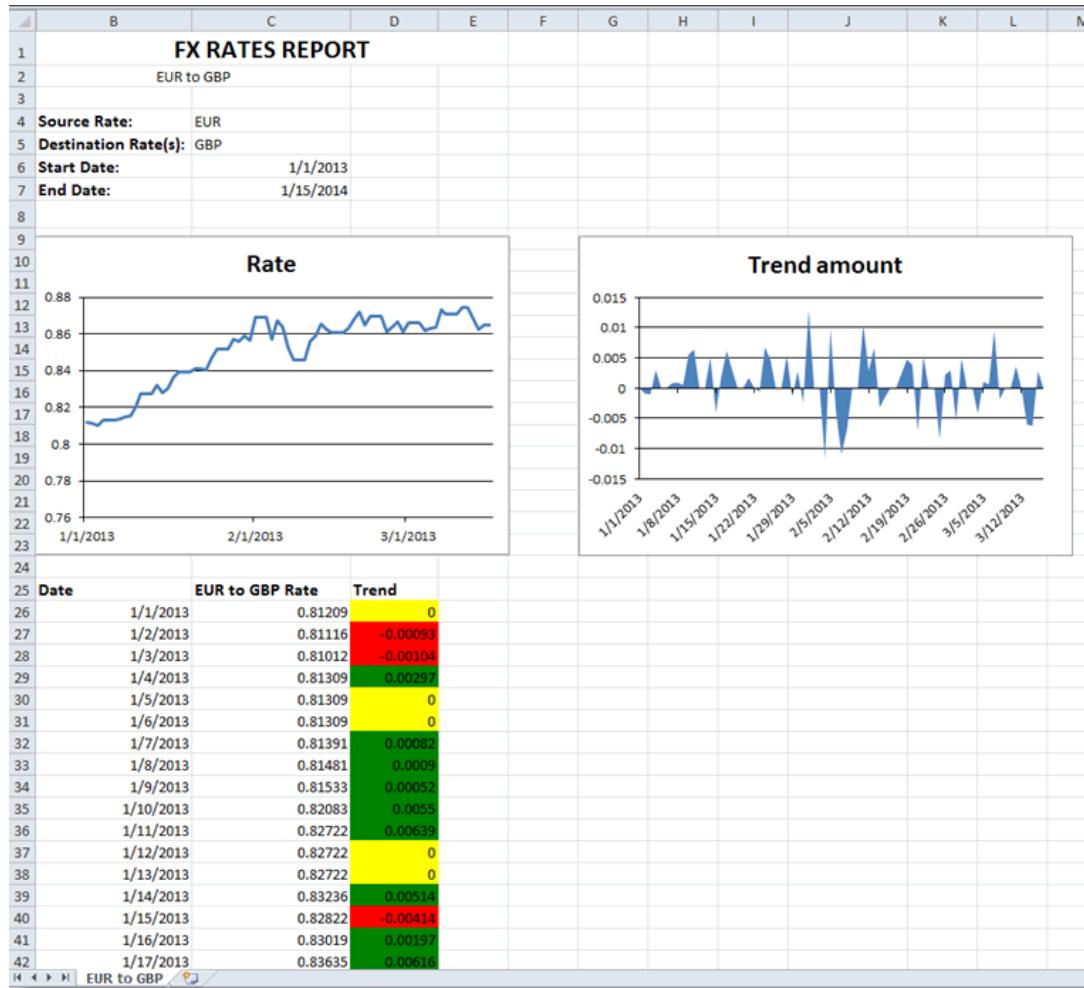
```
//automatically create image names from sheet names, which are named by cross pair  
names
```

```
CHART = ""&DESC&"-Chart 1.jpeg";
```

```
CHART2 = ""&DESC&"-Chart 2.jpeg";
```

```
OUTPUT;  
RUN;  
//Create word document based on table above  
PROC OFFICE FILE="FX Charts.DOC";  
SET CHART_LIST;  
TEMPLATE "&FX Charts_KB.DOC";  
RUN;
```

### 5.13.1.5. Result - Step 1: Excel Workbook with charts



You see that the sheet EUR to GBP contains two graphs. Therefore the intermediary data table used to feed the final WORD report, needs to contain two columns, the first one referring to graph 1 and the second one to graph 2

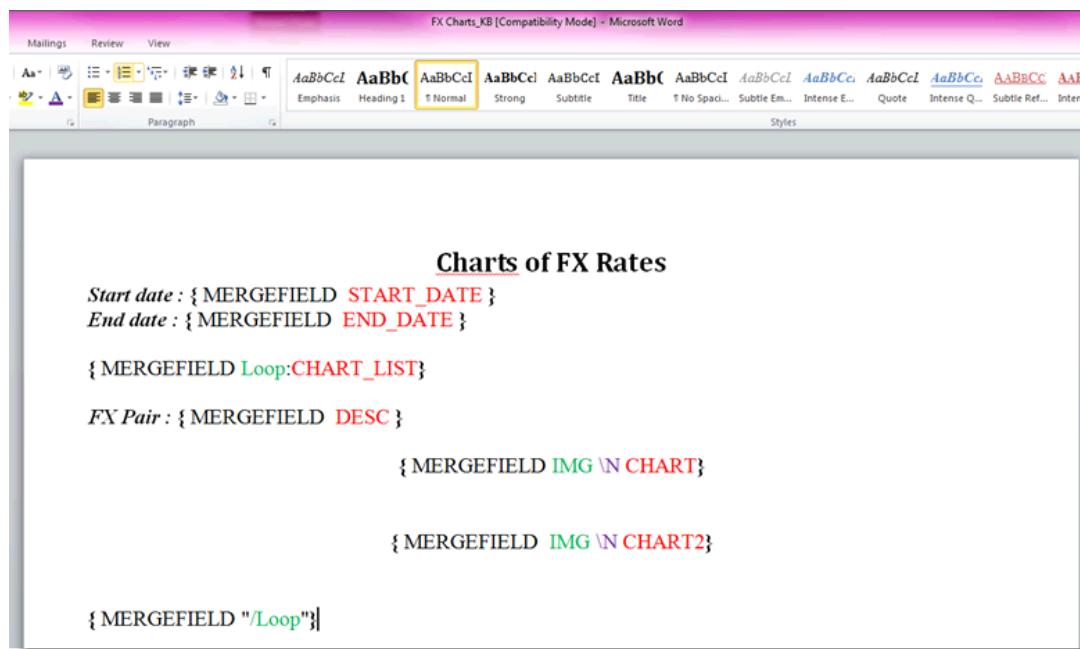
## <Object "CHART\_LIST"

Source Entries mode

```
COLUMN "CURRENCY1" LABEL="CURRENCY1" TYPE="STRING";
COLUMN "DESC";
COLUMN "START_DATE" TYPE="DATE";
COLUMN "END_DATE" TYPE="DATE";
COLUMN "CHART";
COLUMN "CHART2";
```

### 5.13.1.6. WORD Template used to build the WORD report

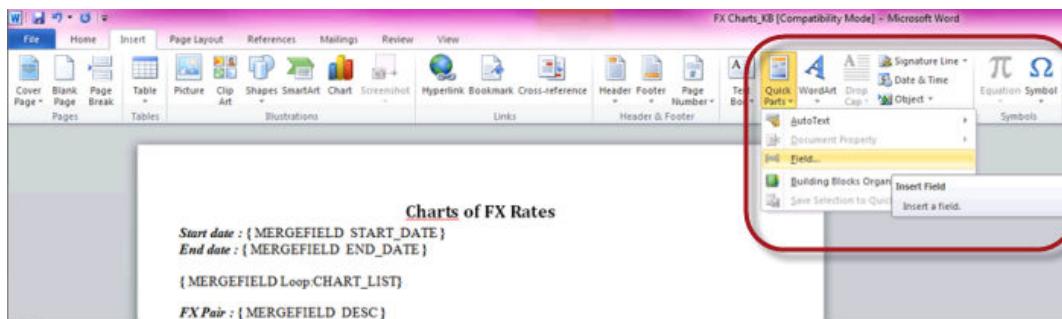
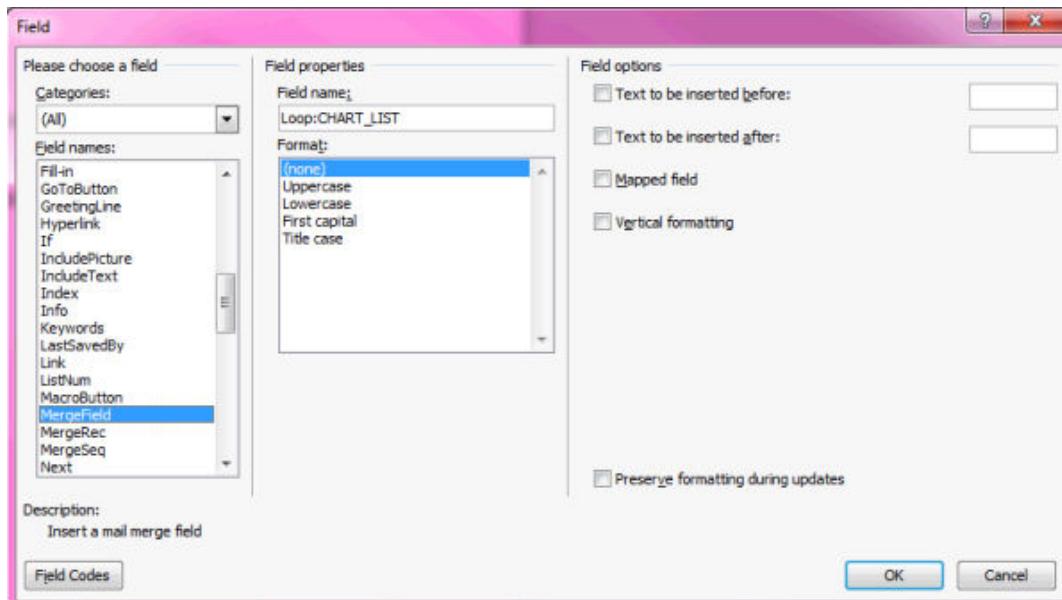
The WORD template used to build the final WORD report, and populated by the CHART\_LIST table looks like this:



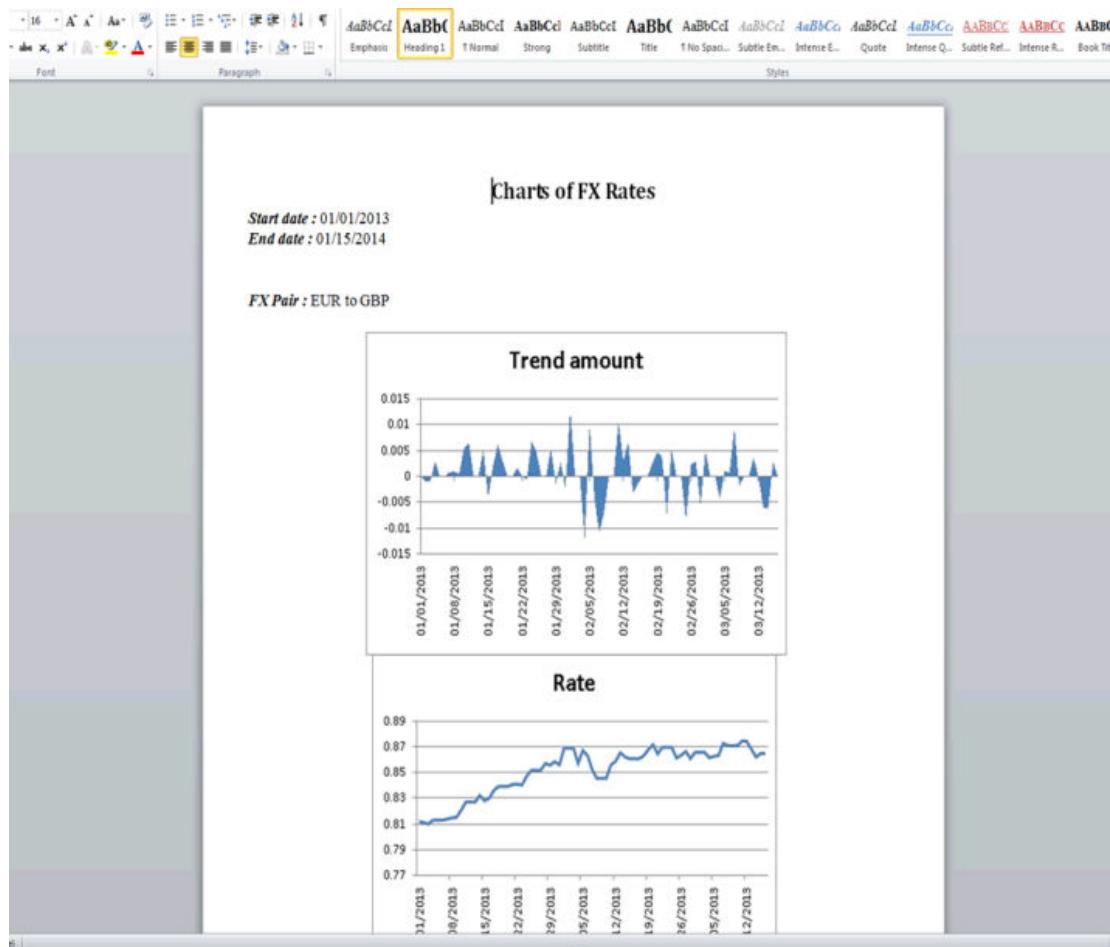
Legend:

- Red labels refer to column names defined in the data table **CHART\_LIST**.
- Green labels refer to mergefield keywords
- Purple labels refer to mergefield options.

To access the mergefield editor in WORD, you need to activate the **Quick Parts**:



### 5.13.1.7. Result - Final WORD report



### 5.13.1.8.

## 5.13.2. PROC EXPORTCHART - Generate graphs and integrate them into WORD docs

### 5.13.2.1. Goal

Generate graphs in an EXCEL file, extract them as JPEG files and integrate them into a WORD document.

### 5.13.2.2. Features being illustrated

- PROC EXPORTCHART step

### 5.13.2.3. Program 1

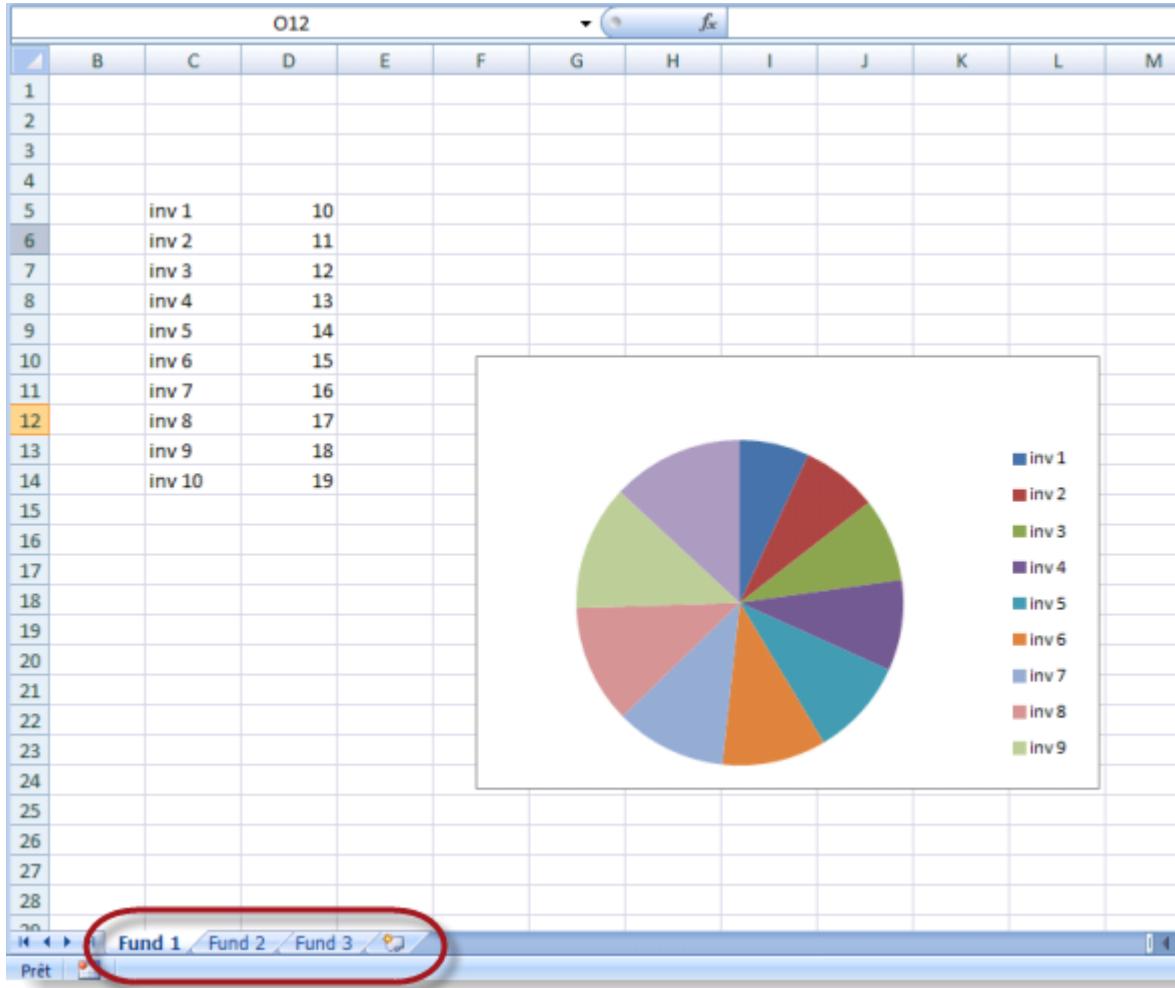
/Generate EXCEL file which contains a set of graphs

```
LIBNAME USER ".";  
  
DATA GRAPH_DATA;  
  
COLUMN FUND;  
  
COLUMN INV;  
  
COLUMN AMOUNT TYPE=FLOAT;  
  
FUND='Fund 1'; INV='inv 1'; AMOUNT=10;OUTPUT;  
  
FUND='Fund 1'; INV='inv 2'; AMOUNT=11;OUTPUT;  
  
FUND='Fund 1'; INV='inv 3'; AMOUNT=12;OUTPUT;  
  
FUND='Fund 1'; INV='inv 4'; AMOUNT=13;OUTPUT;  
  
FUND='Fund 1'; INV='inv 5'; AMOUNT=14;OUTPUT;  
  
FUND='Fund 1'; INV='inv 6'; AMOUNT=15;OUTPUT;  
  
FUND='Fund 1'; INV='inv 7'; AMOUNT=16;OUTPUT;  
  
FUND='Fund 1'; INV='inv 8'; AMOUNT=17;OUTPUT;  
  
FUND='Fund 1'; INV='inv 9'; AMOUNT=18;OUTPUT;  
  
FUND='Fund 1'; INV='inv 10'; AMOUNT=19;OUTPUT;  
  
FUND='Fund 2'; INV='inv 1'; AMOUNT=15;OUTPUT;  
  
FUND='Fund 2'; INV='inv 2'; AMOUNT=16;OUTPUT;
```

```
FUND='Fund 2'; INV='inv 3'; AMOUNT=17;OUTPUT;  
FUND='Fund 2'; INV='inv 4'; AMOUNT=18;OUTPUT;  
FUND='Fund 2'; INV='inv 5'; AMOUNT=19;OUTPUT;  
FUND='Fund 2'; INV='inv 6'; AMOUNT=20;OUTPUT;  
FUND='Fund 2'; INV='inv 7'; AMOUNT=21;OUTPUT;  
FUND='Fund 3'; INV='inv 1'; AMOUNT=10;OUTPUT;  
FUND='Fund 3'; INV='inv 2'; AMOUNT=11;OUTPUT;  
FUND='Fund 3'; INV='inv 3'; AMOUNT=12;OUTPUT;  
FUND='Fund 3'; INV='inv 4'; AMOUNT=13;OUTPUT;  
FUND='Fund 3'; INV='inv 5'; AMOUNT=14;OUTPUT;  
FUND='Fund 3'; INV='inv 6'; AMOUNT=15;OUTPUT;  
FUND='Fund 3'; INV='inv 7'; AMOUNT=16;OUTPUT;  
FUND='Fund 3'; INV='inv 8'; AMOUNT=17;OUTPUT;  
FUND='Fund 3'; INV='inv 9'; AMOUNT=18;OUTPUT;  
RUN;  
  
LIBNAME TEMP1 "\{PRIVATE}\StepStone";  
LIBNAME TEMP2 "\{PRIVATE}\StepStone";  
DATA WORK.[GRAPH_DATA];  
SET TEMP1.[GRAPH_DATA];  
RUN;  
  
PROC MEANS DATA=TEMP2.[GRAPH_DATA] OUT=WORK.[FUNDLIST];
```

```
CLASS FUND;  
RUN;  
  
PROC OFFICE FILE="Generate graph.XLSX";  
  
SET WORK.[GRAPH_DATA] WORK.[FUNDLIST];  
  
TEMPLATE "&Generate graph.XLSX";  
  
RUN;  
  
PROC EXPORTCHART FILE="Generate graph.XLSX";  
  
RUN;  
  
DATA CHART_LIST;  
  
COLUMN FUND;  
  
COLUMN CHART;  
  
COLUMN AMOUNT TYPE=FLOAT;  
  
FUND='Fund 1'; CHART='Fund 1-Chart 4.jpeg';OUTPUT;  
  
FUND='Fund 2'; CHART='Fund 2-Chart 4.jpeg';OUTPUT;  
  
FUND='Fund 3'; CHART='Fund 3-Chart 4.jpeg';OUTPUT;  
  
RUN;
```

### 5.13.2.4. Result 1



### 5.13.2.5. PROGRAM 2

/Create an **eFront Report** document, which generates a WORD document integrating the previously generated graphs

```
%INCLUDE "generate graph";
```

```
LIBNAME TEMP1 "\MDUVINAGE(Private)\StepStone";
```

```
LIBNAME TEMP2 "\MDUVINAGE(Private)\StepStone";
```

```

DATA WORK.[CHART_LIST];
SET TEMP1.[CHART_LIST];
RUN;

DATA WORK.[GRAPH_DATA];
SET TEMP2.[GRAPH_DATA];
RUN;

PROC OFFICE FILE="Charts in Word.DOC";
SET WORK.[CHART_LIST] WORK.[GRAPH_DATA];
TEMPLATE "&Charts in Word.DOC";
RUN;

```

## 5.13.2.6. RESULT 2

[\*\*<FrontReport\*\*](#)

[Current folder](#)

Objects	+table	+New view	+program	+report	+stylesheet	+document	Refresh
FrontReport	<input checked="" type="checkbox"/>						
MDUVINAGE(Private)	<input type="checkbox"/>						
Programs	<input type="checkbox"/>						
Reports	<input type="checkbox"/>						
StepStone	<input type="checkbox"/>						
Stylesheets	<input type="checkbox"/>						
Parent repository							
CHART_LIST							
Charts in Word							
Generate graph							
GRAPH_DATA							

## «Object “Charts in Word”

Editor | Template | Source | HTML output

Editor Execute

Tables Add Delete selected tables

Table	Type	Renamed as	Pathname (design)	Pathname (runtime)	Group by	Filters	
<input type="checkbox"/> CHART_LIST	Table		\IMDUVINAGE(Private)\StepStone		Group by	Filters	
<input type="checkbox"/> GRAPH_DATA	Table		\IMDUVINAGE(Private)\StepStone		Group by	Filters	

Document settings

Include

Document names (runtime)

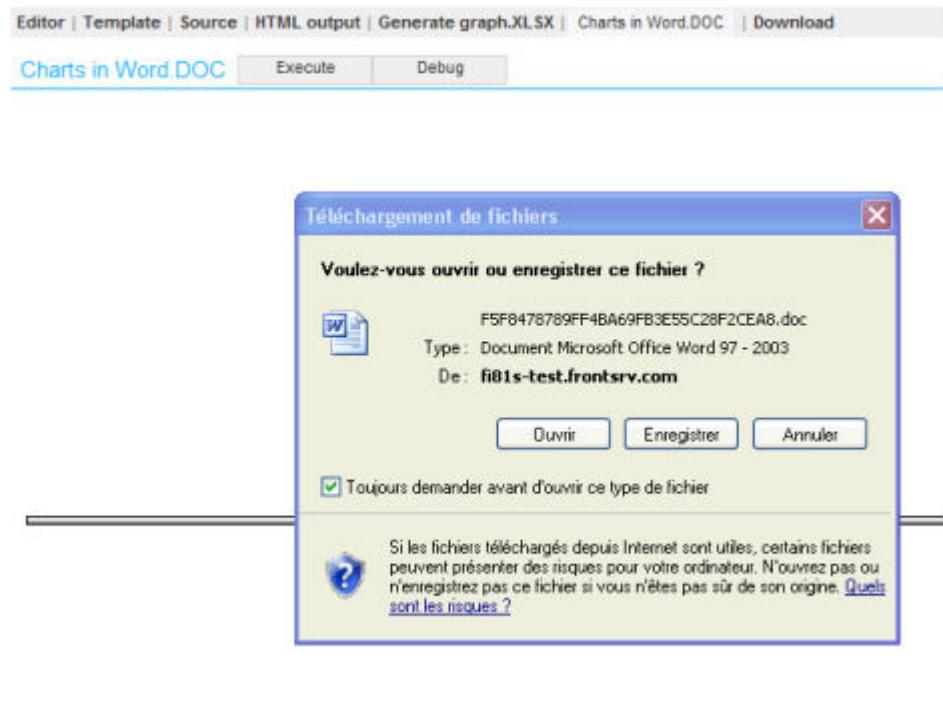
Save in directory

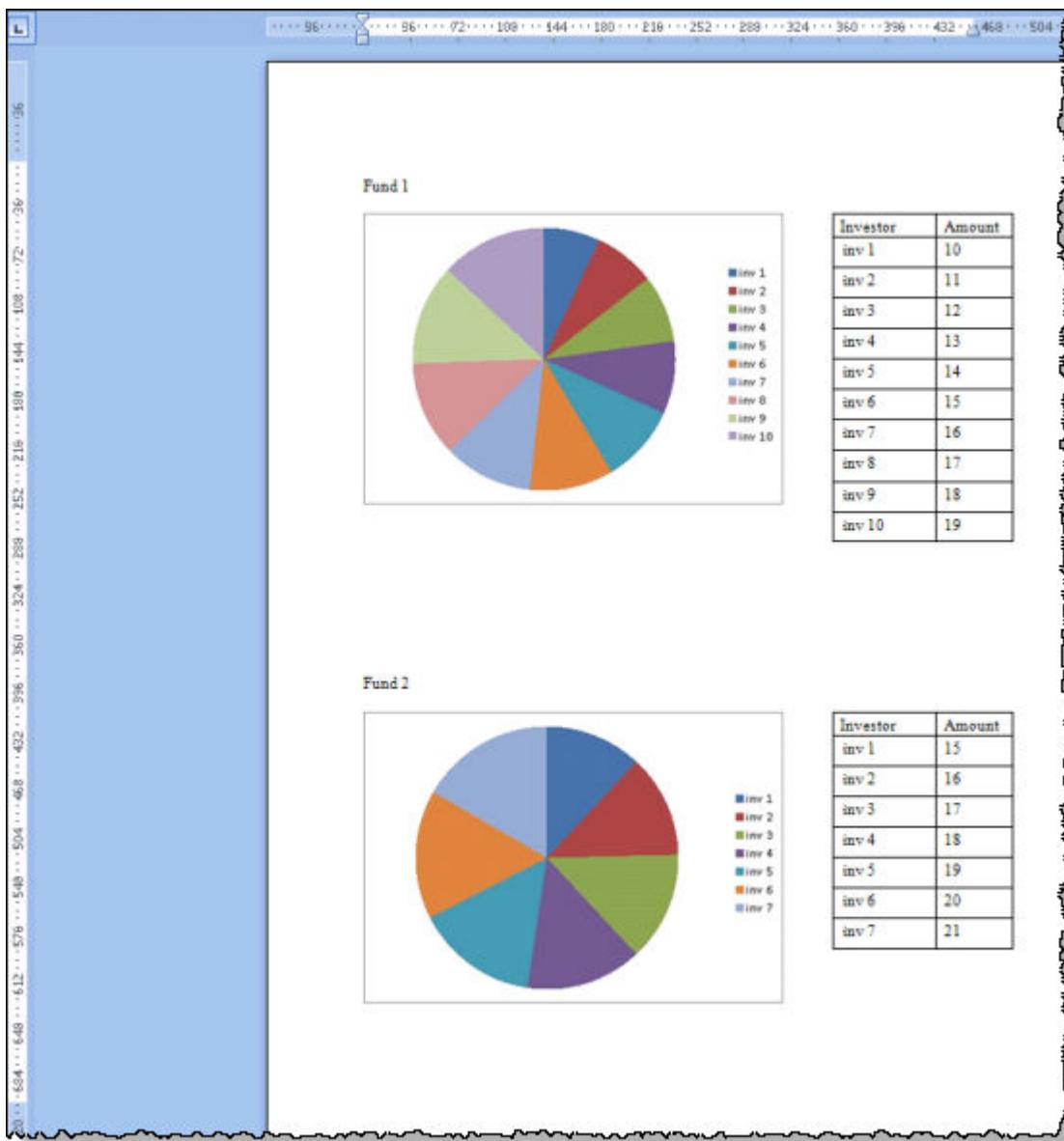
Generate PDF files

Select document to generate

Log

## Object "Charts in Word"





## 5.14. Examples - PROC EXPORTEXCEL step

The following examples illustrate the PROC EXCELEXPORT step, and subordinate statements:

- [PROC EXPORTEXCEL \(1\)](#)

### 5.14.1. PROC EXPORTEXCEL - GETTEMPPATH()

#### 5.14.1.1. Goal

Send a data table from eFront Invest to an EXCEL sheet.

#### 5.14.1.2. Features being illustrated

- [PROC EXPORTEXCEL step](#)
- [Functions - special](#)

#### 5.14.1.3. Program

```
//create a macro-variable to store the complete path to the XLSX file on the client  
%LET NETPATH = GettempPath() & "table_output.xlsx";  
  
//send data table from FI to an EXCEL sheet  
  
PROC EXPORTEXCEL FILE = CSTR(%NETPATH) DROP;  
  
TABLE = work.T_HV_FUND_USERCURRELIGHT  
  
SHEETNAME = "FIA_T_FUND_TRANSACTIONS"  
  
LABEL;  
  
RUN;
```

## 5.15. Examples - PROC FALIBRARY step

The following examples illustrate the PROC FALIBRARY step:

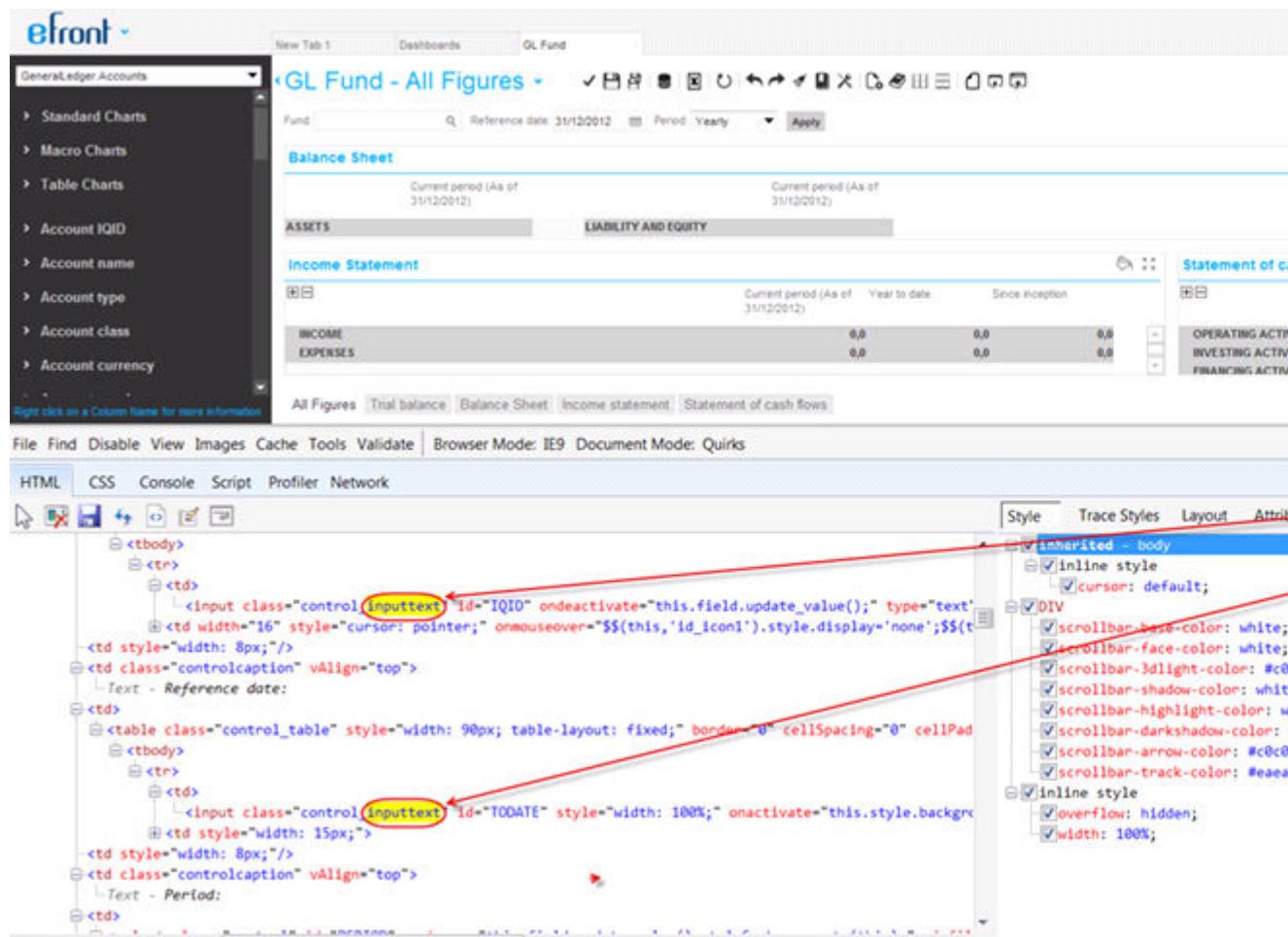
- [PROC FALIBRARY - CFFPortfolio.PortfolioCashflows](#)
- [PROC FALIBRARY - HFFundAssets.Liquidity](#)

### 5.15.1. Tip

► [Tip: How to get the internal names of library parameters](#)

#### Prerequisite:

- You use IE as browser.
1. Open a dashboard, which uses the library you want to call from the FALIBRARY step.
  2. Access the browser's developer tool:
    - Click F12.  
The developer's tool opens.
  3. Search for technical parameter names:
    - a. If the parameter expects text to be entered, do a research on **inputtext**.

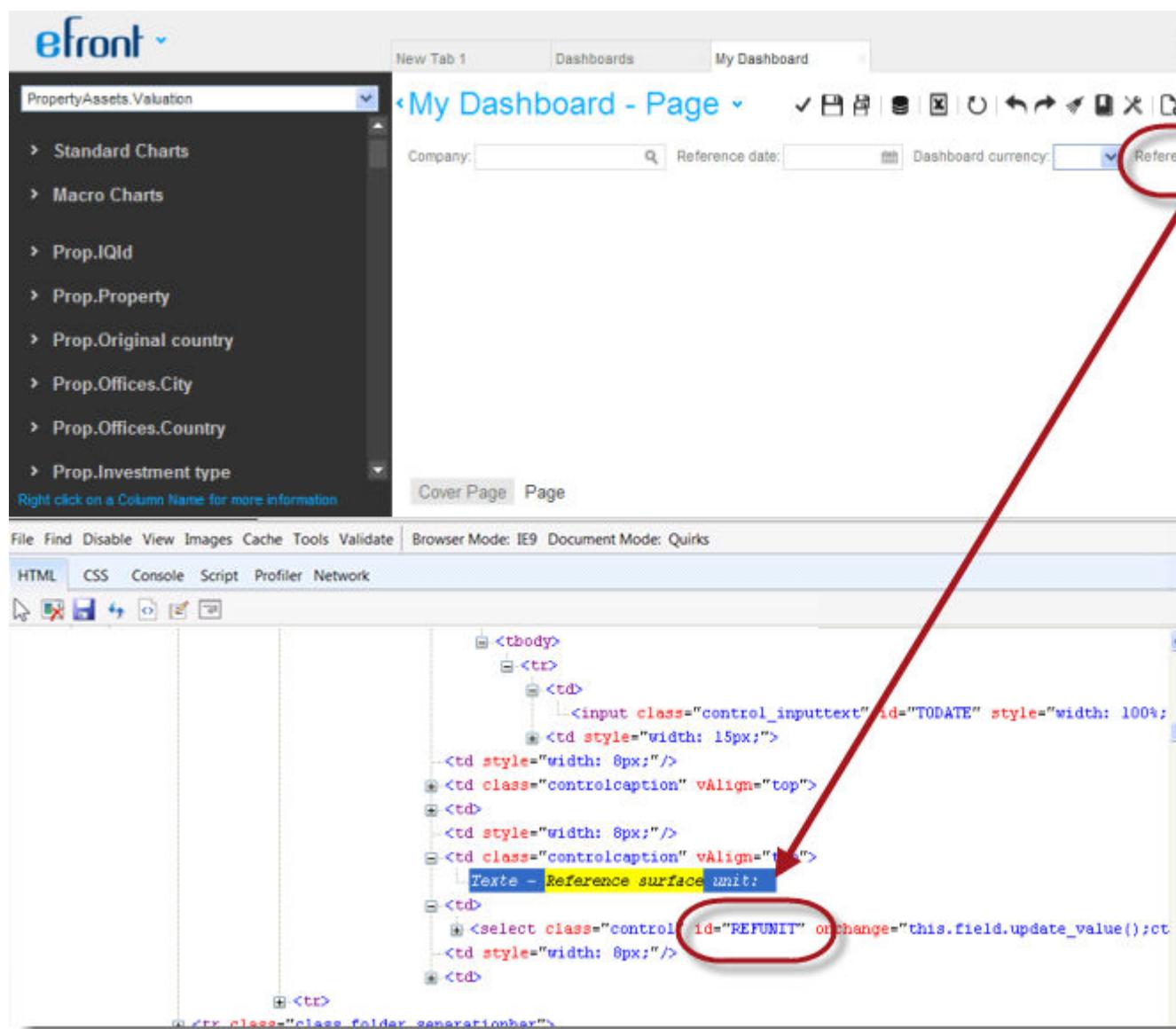


The property **Id** is set to the internal name of the parameter.

- If the parameter expects a list item to be selected, do a research on the value of the selected item.

The screenshot shows the eFront web interface for managing financial figures. At the top, there's a navigation bar with tabs like 'New Tab 1', 'Dashboards', and 'GL Fund'. Below the navigation is a search bar with fields for 'Fund' and 'Reference date' (set to 31/12/2012), and a dropdown for 'Period' with 'Yearly' selected. A red circle highlights this selection. To the right of the search bar is an 'Apply' button. The main content area displays two financial statements: 'Balance Sheet' and 'Income Statement'. The 'Income Statement' table has columns for 'Current period (As of 31/12/2012)', 'Year to date', and 'Since inception'. Below these statements are links for 'All Figures', 'Trial balance', 'Balance Sheet', 'Income statement', and 'Statement of cash flows'. At the bottom of the page, a standard browser menu is visible, followed by developer tool tabs: 'HTML', 'CSS', 'Console', 'Script', 'Profiler', and 'Network'. The 'HTML' tab is active, showing the DOM structure. A red circle highlights the 'Id' attribute of the select element in the HTML code. A red arrow points from this highlighted attribute to the 'Yearly' option in the dropdown menu, which is also highlighted with a red circle.

The property **Id** is set to the internal name of the parameter.  
Here is another example:



## 5.15.2. PROC FALIBRARY - CFFPortfolio.PortfolioCashflows

### 5.15.2.1. Goal

Get the data from the standard library **CFFPortfolio.PortfolioCashflows**.

### 5.15.2.2. Features being illustrated

- PROC FALIBRARY step

### 5.15.2.3. Program

PROC FALIBRARY

OUT=WORK.T\_CASHFLOW\_FORECAST

TABLE="CFFPortfolio.PortfolioCashflows" (IQID=%PORTFOLIO\_ID  
TODATE=%ACTUAL\_END\_DATE);

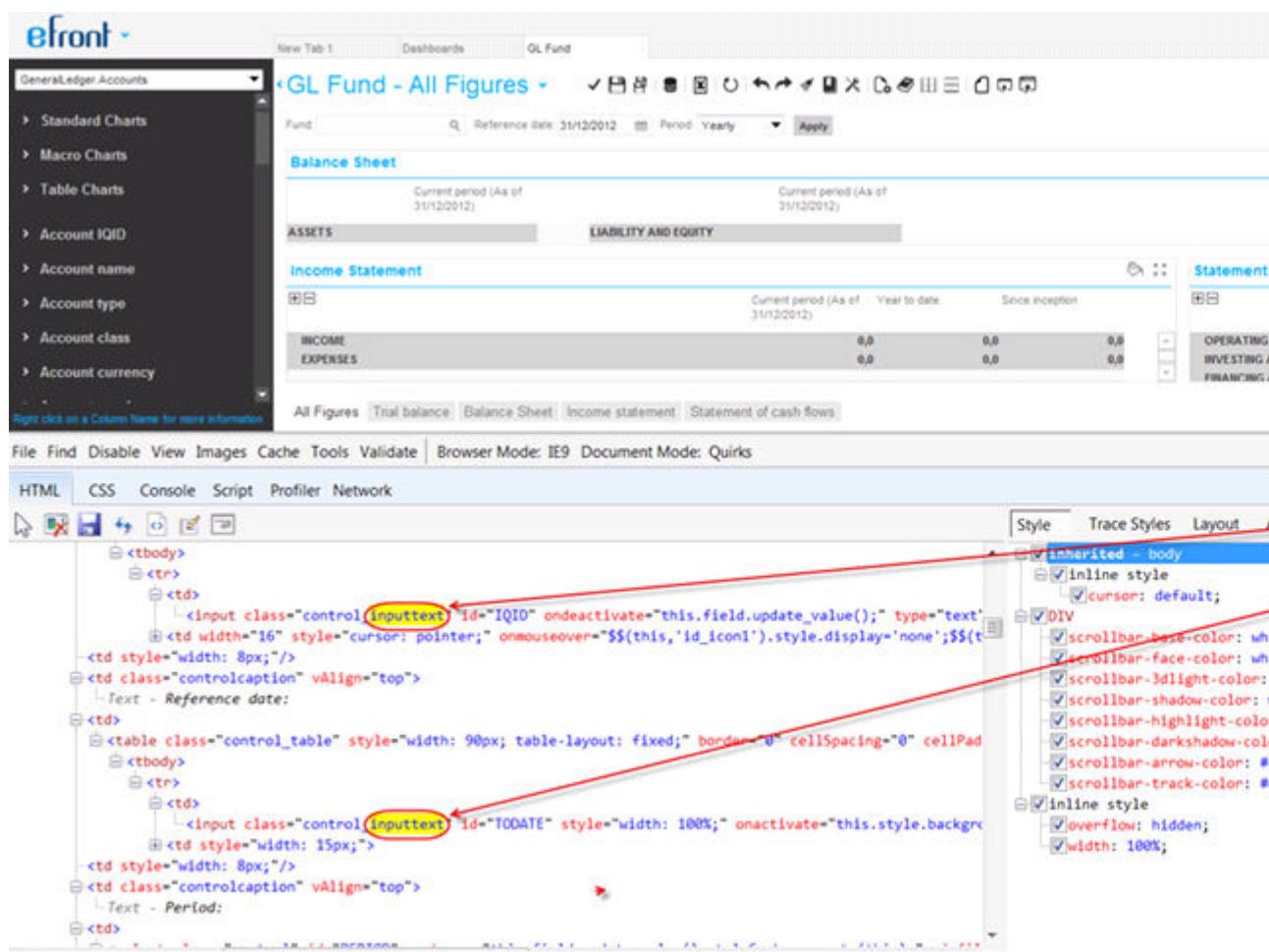
RUN;

### 5.15.2.4. Comments

- The key in using the FALIBRARY step correctly is to find the parameters to be passed along together with the library. When you refer to the parameters to be passed along with the library, you need to use the technical (internal names) of the parameters. Get a [► Tip: How to get the internal names of library parameters.](#)

#### Prerequisite:

- You use IE as browser.
  1. Open a dashboard, which uses the library you want to call from the FALIBRARY step.
  2. Access the browser's developer tool:
    - Click F12.  
The developer's tool opens.
  3. Search for technical parameter names:
    - a. If the parameter expects text to be entered, do a research on **inputtext**.



The property **Id** is set to the internal name of the parameter.

- If the parameter expects a list item to be selected, do a research on the value of the selected item.

The screenshot shows the eFront interface for managing financial figures. At the top, there's a navigation bar with tabs like 'New Tab 1', 'Dashboards', and 'GL Fund'. On the left, a sidebar titled 'General Ledger Accounts' lists various account categories. The main content area displays a 'Balance Sheet' and an 'Income Statement' for the 'GL Fund - All Figures' period, ending on '31/12/2012'. The 'Period' dropdown in the header is set to 'Yearly' and is highlighted with a red circle. A red arrow points from this circle down to the corresponding code in the developer tools.

**HTML Structure:**

```

<tbody>
  <tr>
    <td><input class="control_inputtext" id="TODATE" style="width: 100%; onactivate="this.style.background-color = '#fff'; this.value = this.defaultValue;" type="text" value="31/12/2012" /></td>
    <td style="width: 15px;"></td>
    <td style="width: 8px;"></td>
    <td class="controlcaption" vAlign="top">Text - Period:</td>
    <td>
      <select class="control" id="PERIOD" onchange="this.field.update_value();ctrl_Combobox_execute(this); ajaxfill(this);>
        <option value=""></option>
        <option value="Monthly"></option>
        <option value="Quarterly"></option>
        <option value="Yearly" selected="selected"></option>
        <option value="YearToDate"></option>
      </select>
    <td style="width: 8px;"></td>
  </tr>

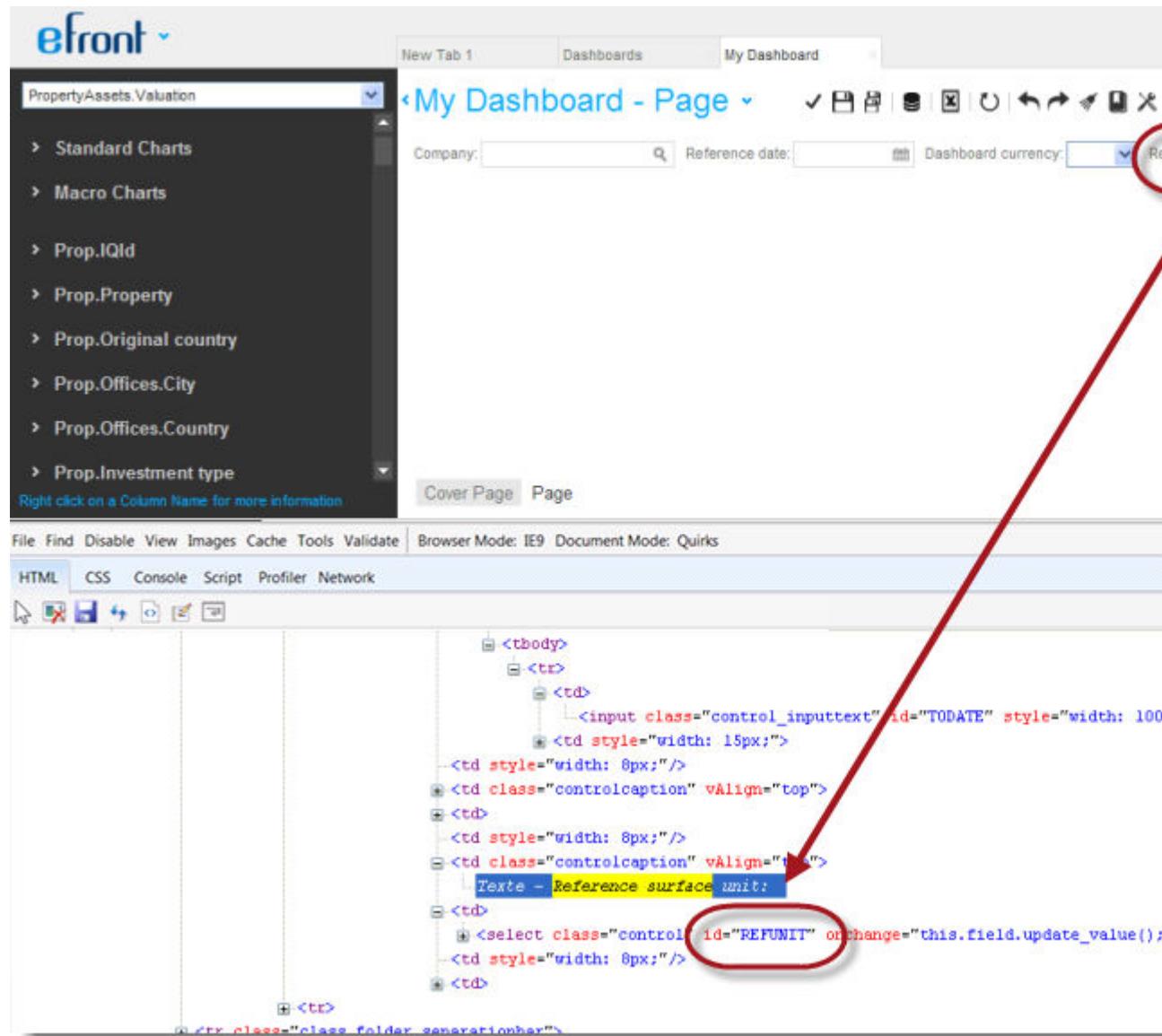
```

**Style Inspector:**

The style inspector on the right shows the CSS inheritance for the selected element. The 'control' class is defined with the following properties:

- inherited - body
- inherited - select
- .control
  - cursor: default
  - color: #333333;
  - font-family: Arial, sans-serif;
  - font-size: 8pt;
  - font-weight: bold;

The property **Id** is set to the internal name of the parameter.  
Here is another example:



- For more details about the parameters to be used with each library, please refer to the **eFront Analytics - Catalogue of Standard Libraries and Sample Dashboards**.

### 5.15.3. PROC FALIBRARY - HFFundAssets.Liquidity

#### 5.15.3.1. Goal

Get the data from the standard library **HFFundAssets.Liquidity**.

### 5.15.3.2. Features being illustrated

- PROC FALIBRARY step

### 5.15.3.3. Program

PROC FALIBRARY

OUT=WORK.HFFUNDASSETS\_LIQUIDITY

TABLE="HFFundAssets.Liquidity(PORTFOLIO)" (IQID=%PORTFOLIO\_ID  
TODATE=%Reference\_Date REFCURRENCY=%Currency);

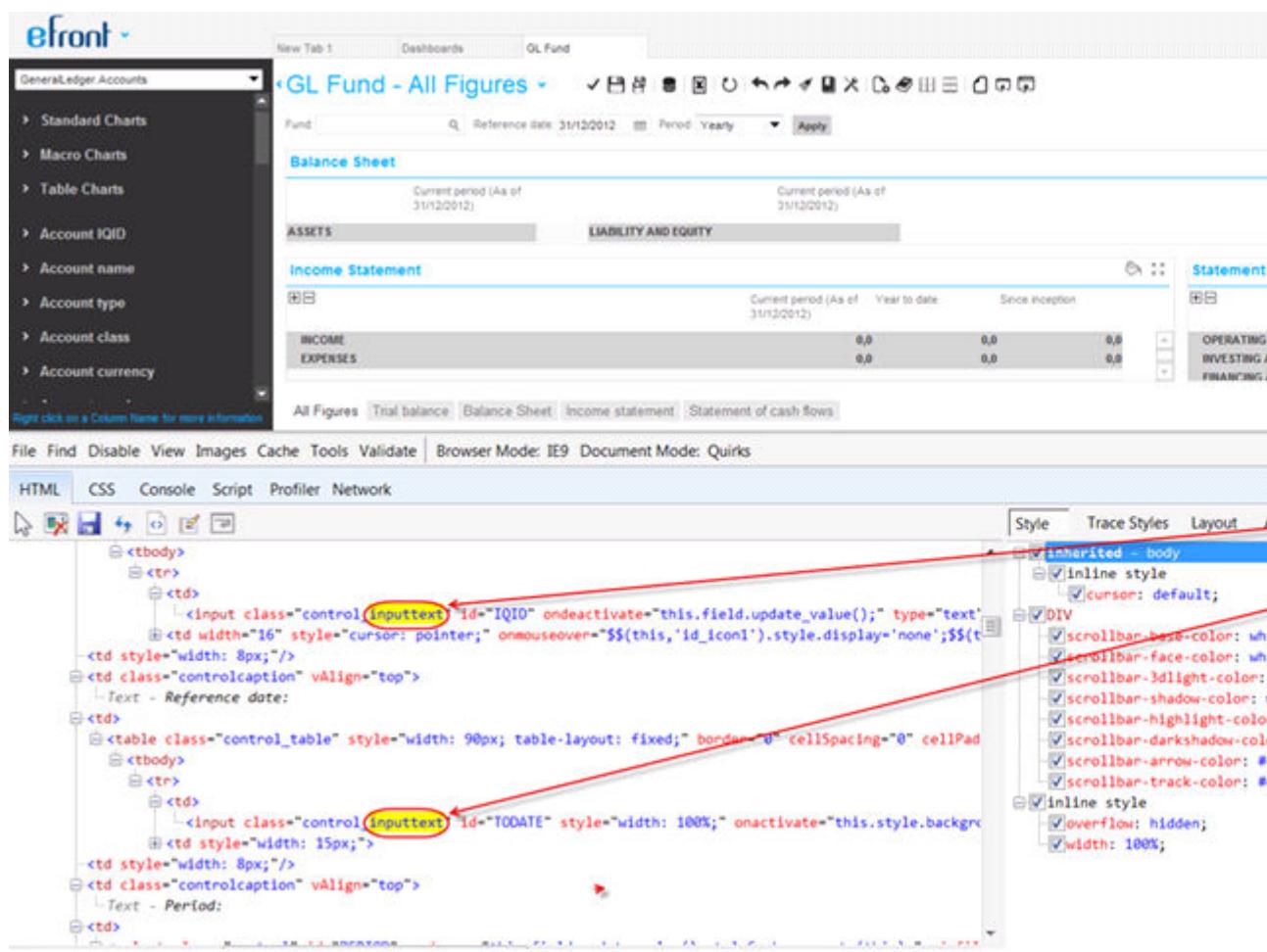
RUN;

### 5.15.3.4. Comments

- When using a library such as the **HFFundAssets.Liquidity**, you need to define the scope of the library, which is done here above by adding (PORTFOLIO) to the library name. In other libraries the element which defines the library scope might be (FUND) for example.
- The key in using the FALIBRARY step correctly is to find the parameters to be passed along together with the library. When you refer to the parameters to be passed along with the library, you need to use the technical (internal names) of the parameters. Get a  [Tip: How to get the internal names of library parameters](#).

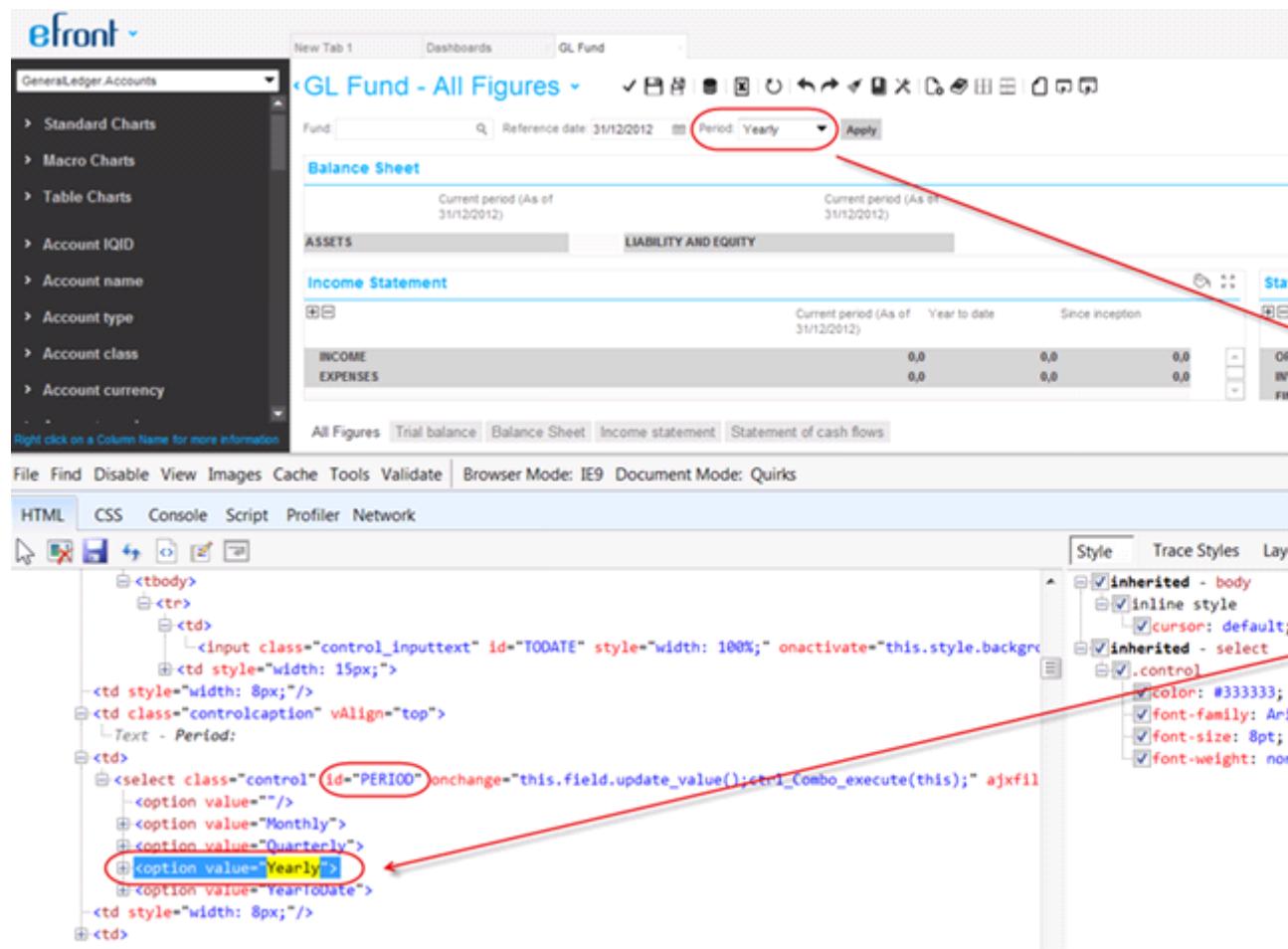
#### Prerequisite:

- You use IE as browser.
1. Open a dashboard, which uses the library you want to call from the FALIBRARY step.
  2. Access the browser's developer tool:
    - Click F12.  
The developer's tool opens.
  3. Search for technical parameter names:
    - a. If the parameter expects text to be entered, do a research on **inputtext**.

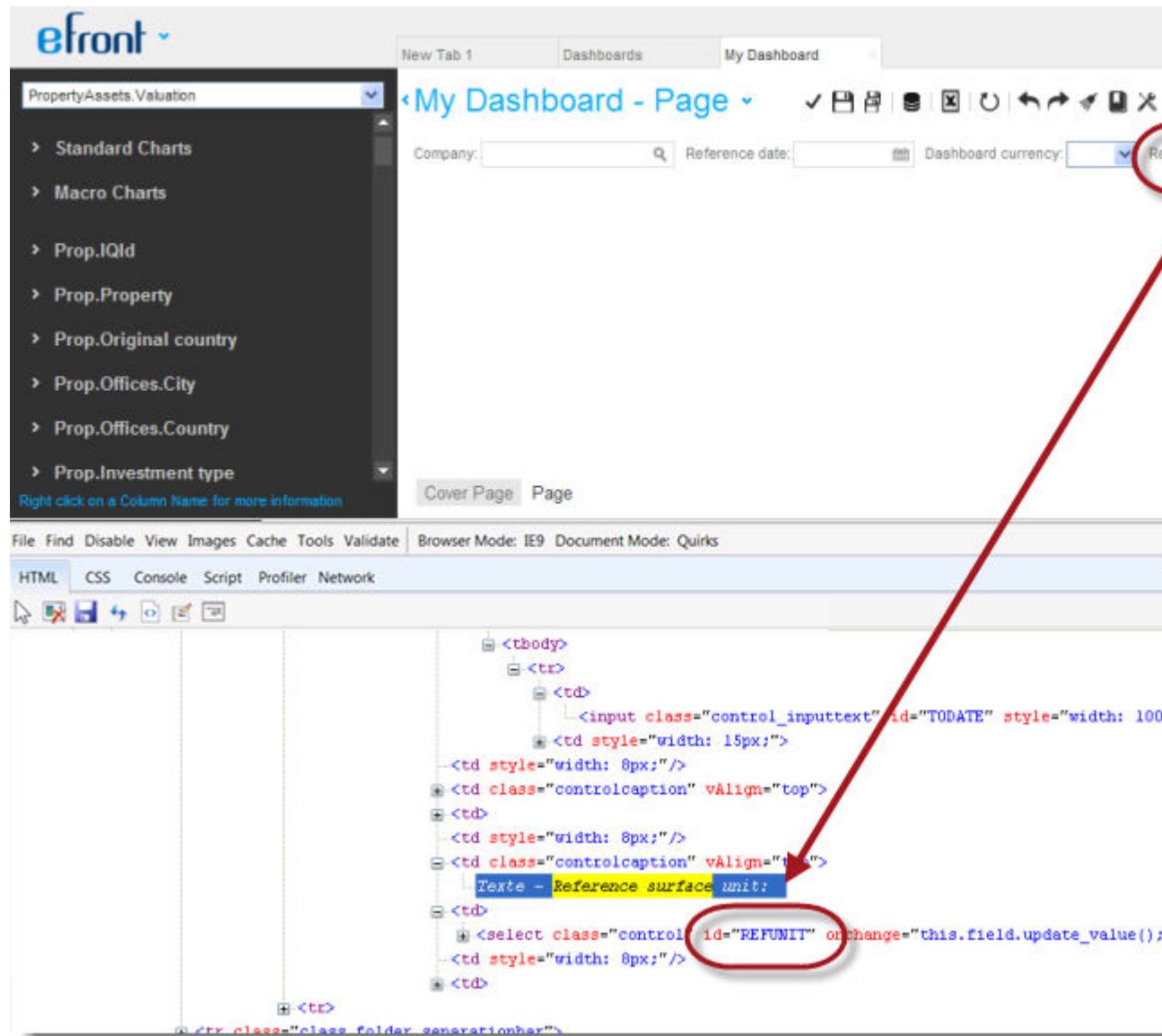


The property **Id** is set to the internal name of the parameter.

- If the parameter expects a list item to be selected, do a research on the value of the selected item.



The property **Id** is set to the internal name of the parameter.  
Here is another example:



- For more details about the parameters to be used with each library, please refer to the **eFront Analytics - Catalogue of Standard Libraries and Sample Dashboards**.

### 5.15.3.5. Parameters - How to get the internal names of library parameters

**Prerequisite:**

- You use IE as browser.
1. Open a dashboard, which uses the library you want to call from the FALIBRARY step.
  2. Access the browser's developer tool:
    - Click F12.  
The developer's tool opens.
  3. Search for technical parameter names:
    - a. If the parameter expects text to be entered, do a research on **inputtext**.

The screenshot shows the eFront web interface for 'GL Fund - All Figures'. On the left is a sidebar with navigation links like 'Standard Charts', 'Macro Charts', etc. The main area displays a 'Balance Sheet' and 'Income Statement' for the period ending 31/12/2012. Below these are tabs for 'All Figures', 'Trial balance', 'Balance Sheet', 'Income statement', and 'Statement of cash flows'. At the bottom, the developer tools are open, showing the HTML structure. Two specific input fields are circled in red and have arrows pointing to their 'Id' attributes in the CSS inspector on the right. The first circled field has the id 'IQID' and the second has the id 'TODATE'.

The property **Id** is set to the internal name of the parameter.

- b. If the parameter expects a list item to be selected, do a research on the value of the selected item.

The screenshot shows the eFront web interface for managing financial figures. At the top, there's a navigation bar with tabs like 'New Tab 1', 'Dashboards', and 'GL Fund'. Below the navigation is a search bar and a dropdown for 'Fund'. To the right of the fund dropdown is a 'Period' dropdown set to 'Yearly', which is circled in red. Below the search bar, there are two main sections: 'Balance Sheet' and 'Income Statement'. The 'Income Statement' section has columns for 'Current period (As of 31/12/2012)', 'Year to date', and 'Since inception'. At the bottom of the page, there's a menu bar with options like 'File', 'Find', 'Disable', 'View', 'Images', 'Cache', 'Tools', 'Validate', 'Browser Mode: IE9', and 'Document Mode: Quirks'. The developer tools are open at the bottom, showing the HTML structure of the page. The 'HTML' tab is selected. In the DOM tree, a red circle highlights the 'id="PERIOD"' attribute of the select element. Another red circle highlights the 'Yearly' option in the dropdown menu. The developer tools also show the CSS styles applied to the page, including styles for the body, select elements, and specific class names like '.control'.

The property **Id** is set to the internal name of the parameter.  
Here is another example:

The screenshot shows the eFront dashboard interface with a sidebar containing navigation items like Standard Charts, Macro Charts, Prop.IQId, Prop.Property, Prop.Original country, Prop.Offices.City, Prop.Offices.Country, and Prop.Investment type. The main area is titled 'My Dashboard - Page' and includes search fields for Company, Reference date, and Dashboard currency. A red arrow points from the top right towards the bottom right corner where the element tree is displayed. The element tree shows the HTML structure of a table row, specifically focusing on a select element with id="REFUNIT".

```
<tbody>
  <tr>
    <td>
      <input class="control_inputtext" id="TODATE" style="width: 100%; height: 25px;" type="text"/>
    </td>
    <td style="width: 15px;">
      <input type="button" value="..."/>
    </td>
    <td class="controlcaption" vAlign="top">
      <td>
        <td style="width: 8px;">
          <input type="button" value="..."/>
        </td>
        <td class="controlcaption" vAlign="top">
          Texte - Reference surface unit:
        </td>
      </td>
      <td>
        <select class="control" id="REFUNIT" onchange="this.field.update_value();ctc_update(this.field, this.value);"/>
      </td>
      <td style="width: 8px;">
        <input type="button" value="..."/>
      </td>
    </td>
  </tr>
<tr class="class_folder_separationbar">
```

## 5.16. Examples - PROC FAQUERY step

The following examples illustrate the PROC FAQUERY step:

- [PROC FAQUERY - Contacts - EntireTable](#)
- [PROC FAQUERY - %LET](#)

### 5.16.1. PROC FAQUERY - Contacts - EntireTable

#### 5.16.1.1. Goal

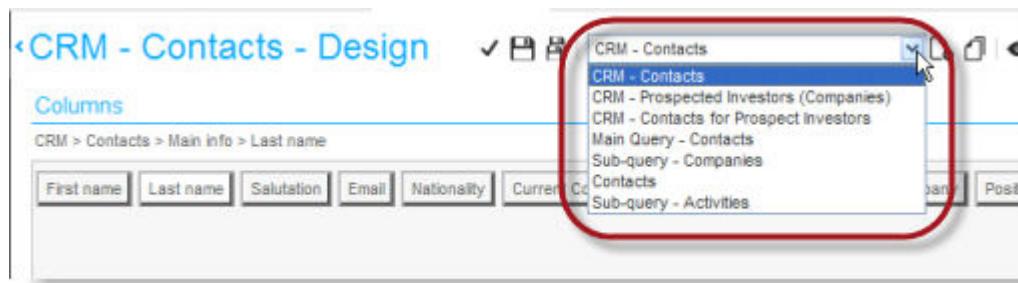
Integrate the results of a QueryBuilder query requiring an input parameter into an eFront Script program, and build a global cube.

#### 5.16.1.2. Features being illustrated

- [PROC FAQUERY step](#)

#### 5.16.1.3. QueryBuilder query

We have a query component entitled **CRM - Contacts**. This query gathers several queries.



We are interested in the query entitled: **CRM - Contacts**:

The screenshot shows the eFront CRM - Contacts - Design interface. On the left, there's a sidebar with 'Objects' and 'Properties' sections. Under 'Properties', 'Last name' is selected, showing its main settings like Main, Alias, Layout, Caption, Format, Sort, Width, Hidden, Total, Force Picklist, and Force Raw Value. The main area is titled 'CRM - Contacts - Design' and shows 'Columns' (First name, Last name, Salutation, Email, Nationality, Current Company, LinkedIn, Mobile, Previous Company, Position, Date) and 'Filters'. At the bottom, there are 'Design' and 'Overview' tabs, with 'Design' being active.

The result being generated by the query looks as follows:

CRM - Contacts - Overview									
CRM - Contacts									
To group according to a column, drag and drop its header here									
First name	Last name	Salutation	Email	Primary Citizenship	Current Company	LinkedIn	Mobile	Prev	Next
			aauclare@efront...	France					
			mduvinage@efron...	France					
Jeff	ALTMAN	Mr	jaltman@web.com	United States of America					
Alexandra	Bonono		ABonono@efront...	France					
Alexandre	BOUILLON	Mr	alexandre.bouillon...	France					
Olivier	DELLENBACH	Mr		France	eFront				eFront
Jeanne	DEREK	Mrs	jeanne.derek@co...	Germany					
Violette	François	Mr	fviolette@efront.c...	France	eFront				
Adrien	Hunot		AHunot@efront.c...	France					
	Investor 3			France					
	Investor 5			France					
Catherine	JONES	Mrs	cat.jones@front.ca	Canada	Company D				Corr
Duvinage	Monika		mduvinage@efron...	France					+33669444699
Gilles	ROBERT	Mr	gilles.robert@gag...	France					
Baily	Séverine	Mrs	sbaily@efront.com	France	eFront				
Adam	Smith	Mr		United States of America	Company C				Corr
John	Smith	Mr	jsmith@efront.com	France					(+33) 6 66 66 66 66
	Test Workflow			France					
Emmanuel	Trouve			France					
François	Violette	Mr	fviolette@efront.c...	France	Company B				
	XX_DIRECTOR_BANK_E...	Mrs		France	XX_INVESTOR_B...				
	XX_INVESTOR_CONTACT...	Mr		France	XX_COMPANY_P...				

### 5.16.1.4. Program

```
LIBNAME USER ":";
```

```
PROC FAQUERY QUERY="FIA405 - QueryBuilderSamples\CRM - Contacts";
```

```
TABLE "CRM - Contacts" OUT=ContactsTable;
```

```
RUN;
```

```
PROC PRINT DATA=ContactsTable;
```

RUN;

### 5.16.1.5. Result

### 5.16.1.6.

### 5.16.1.7. Comments

- The program creates by default a global cube.
- You can use more than 1 TABLE statement in a PROC FAQUERY step, which would allow you in the example above for instance, to integrate all the queries supported by the query component into the eFront Script program.

## 5.16.2. PROC FAQUERY - %LET

### 5.16.2.1. Goal

Extract data from a QueryBuilder query, and export it into an eFront Report data table.

### 5.16.2.2. Features being illustrated

- PROC FAQUERY step
- %LET statement

### 5.16.2.3. Program

```
%LET FUND_ID= "0F46B969A46A45F88C921CF55F27C357";
```

```
%LET QUERY_NAME= "SubQuery";
```

```
PROC FAQUERY QUERY="DEALS\TestQB"(IQID=%FUND_ID);
```

```
TABLE "default"(COLUMNS = COL2 COL4) OUT=Work.test;
```

```
TABLE %QUERY_NAME OUT=Work.test2;
```

RUN;

```
PROC PRINT DATA=WORK.TEST;
```

RUN;

```
PROC PRINT DATA=WORK.TEST2;  
RUN;
```

#### 5.16.2.4. Comments

The example above, exports from the query component entitled **TestQB** and located in the Dashboards folder **DEALS**, two queries, the query entitled **default** and the query entitled **SubQuery**. For the **default** query, only columns 2 and 4 are exported into the eFront Report table **Work.test**.

## 5.17. Examples - PROC FORMAT step

The following examples illustrate the PROC FORMAT step, and subordinate statements:

- PROC FORMAT - COLUMN - FORMAT
- PROC FORMAT - PICTURE
- PROC FORMAT - VALUE (1)
- PROC FORMAT - VALUE (2)
- PROC EXPORT - %LET - PROC FORMAT
- PROC FORMAT - PICTURE - large negative numbers

### 5.17.1. PROC FORMAT - COLUMN - FORMAT

#### 5.17.1.1. Goal

Apply a format to a column.

#### 5.17.1.2. Features being illustrated

- PROC FORMAT step
- FORMAT option

#### 5.17.1.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

//Create a personalized format

PROC FORMAT;

PICTURE P\_Money

LOW-HIGH = '\$ 0000.00';

RUN;

/\*Create a data table from scratch and apply the customized format\*/

```
DATA TEST.CONTRACTS;  
  
COLUMN CONTRACT_ID LABEL="Contract Id" TYPE=INTEGER;  
  
COLUMN USER_ID LABEL="User Id" TYPE=INTEGER;  
  
COLUMN UNIT_PRICE LABEL="Unit Price" TYPE=DOUBLEFORMAT=P_MONEY; ;  
  
COLUMN QTY LABEL="Quantity" TYPE=INTEGER;  
  
//Write the first line of the table  
  
CONTRACT_ID = "593";  
  
USER_ID = "101";  
  
UNIT_PRICE ="4220.1";  
  
QTY ="2";  
  
OUTPUT;  
  
//Write the second line of the table  
  
CONTRACT_ID = "594";  
  
USER_ID = "101";  
  
UNIT_PRICE ="1299";  
  
QTY ="3";  
  
OUTPUT;  
  
RUN;  
  
//Display the table  
  
PROC PRINT DATA = TEST.CONTRACTS;  
  
RUN;
```

## 5.17.1.4. Result

	CONTRACT	USER_ID	UNIT_PRICE	QTY
1	593	101	\$4220,10	2
2	594	101	\$1299,00	3
3	595	101	\$7501,00	1
4	596	102	\$0011,40	100
5	597	102	\$6430,20	1
6	596	103	\$7501,00	3
7	597	101	\$6430,20	5
8	598	105	\$20030,20	5

## 5.17.2. PROC FORMAT - PICTURE

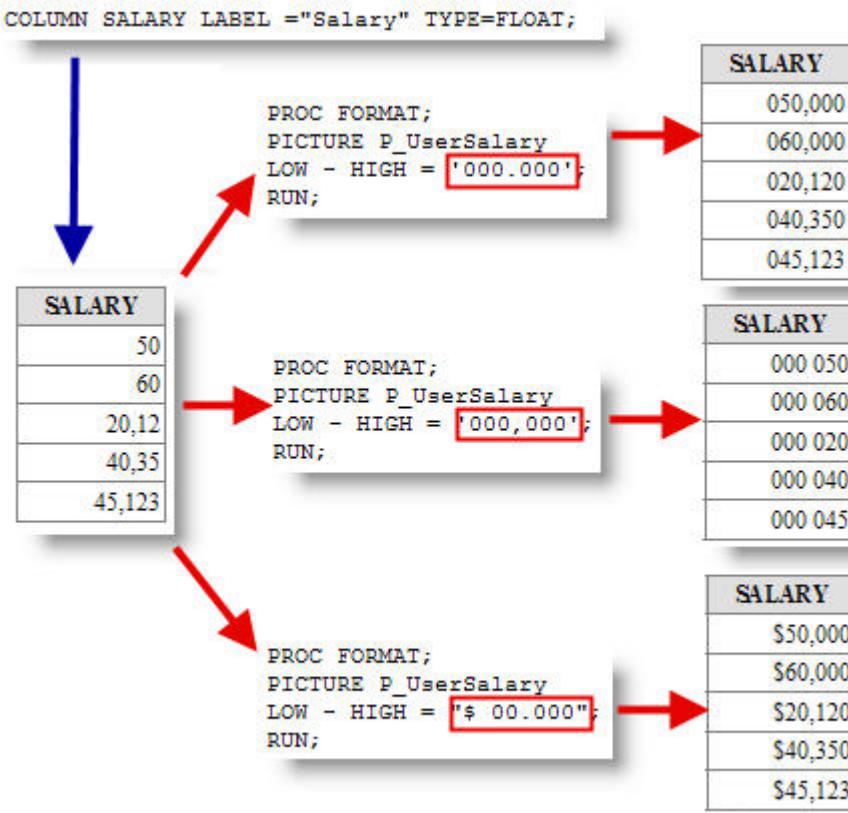
### 5.17.2.1. Goal

Create different pictures to display the same salary figures.

### 5.17.2.2. Features being illustrated

- PROC FORMAT step
- PICTURE statement

### 5.17.2.3. Program



### 5.17.3. PROC FORMAT - VALUE (1)

#### 5.17.3.1. Goal

Define a personalized character format that replaces city names by their abbreviation.

#### 5.17.3.2. Features being illustrated

- PROC FORMAT step
- VALUE statement

#### 5.17.3.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

```
//Create a personalized format  
  
PROC FORMAT;  
  
VALUE $FCITY  
  
"New York" = "NY"  
  
"Paris" = "P"  
  
"Bordeaux" = "B";  
  
RUN;  
  
//Apply the personalized format to variable values  
  
PROC PRINT DATA = TEST.USERS_AND_CONTRACTS;  
  
FORMAT CITY $FCITY.;  
  
RUN;
```

### 5.17.3.4. Result

	USER_ID	FIRSTNAME	LASTNAME	ZIP	CITY
1	101	Bobby	Schmidt	10021	New York
2	101	Bobby	Schmidt	10021	New York
3	101	Bobby	Schmidt	10021	New York
4	101	Bobby	Schmidt	10021	New York
5	102	Ruber	Zodi	75001	Paris
6	102	Ruber	Zodi	75001	Paris
7	103	Sandra	Specker	33000	Bordeaux
8	104	Sandra	Specker	33000	Bordeaux
9	105	Marjorie	Pattern	10021	New York



	USER_ID	FIRSTNAME	LASTNAME	ZIP	CITY
1	101	Bobby	Schmidt	10021	NY
2	101	Bobby	Schmidt	10021	NY
3	101	Bobby	Schmidt	10021	NY
4	101	Bobby	Schmidt	10021	NY
5	102	Ruber	Zodi	75001	P
6	102	Ruber	Zodi	75001	P
7	103	Sandra	Specker	33000	B
8	104	Sandra	Specker	33000	B
9	105	Marjorie	Pattern	10021	NY

## 5.17.4. PROC FORMAT - VALUE (2)

### 5.17.4.1. Goal

Define a personalized character format that replaces value ranges by character strings.

### 5.17.4.2. Features being illustrated

- PROC FORMAT step
- VALUE statement

### 5.17.4.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a personalized format

PROC FORMAT;

VALUE \$FCHILDREN

1 - 2 = "Less than 2 children"

3 = "3 children"

3 - HIGH = "More than 3 children"

OTHER = "Not specified";

RUN;

//Apply the personalized format to values to be displayed

PROC PRINT DATA = TEST.USERS;

FORMAT CHILDREN \$FCHILDREN.;

RUN;

#### 5.17.4.4. Result

	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	CHILDREN
1	101	Bobby	Schmidt	05/03/1959	2
2	102	Ruber	Zodi	25/01/1980	1
3	103	Deborah	Woodpecker	04/11/1954	5
4	104	Sandra	Specker	01/12/1962	4
5	105	Marjorie	Pattern	12/03/1983	



	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	CHILDREN
1	101	Bobby	Schmidt	05/03/1959	Less than 2 children
2	102	Ruber	Zodi	25/01/1980	Less than 2 children
3	103	Deborah	Woodpecker	04/11/1954	More than 3 children
4	104	Sandra	Specker	01/12/1962	More than 3 children
5	105	Marjorie	Pattern	12/03/1983	Not specified

## 5.17.5. PROC FORMAT - PICTURE - large negative numbers

### 5.17.5.1. Goal

Format large negative numbers in the following way:

#### Before formatting

HTML Output Execute										
TION	DATE	FUND_IQID	DESCRIPTION	TOTALINVE	TOTALDISTI	NET_CASHFI	NAV_ADJUST	CUMULATIV	FUND	P
0	06/30/2003	17C0F1E86AEf				- 5783085.8483	5783085.8483	5783085.8483	@@test	
0	06/30/2003	17C0F1E86AEf				- 5783085.8483	5783085.8483	5783085.8483	@@test	
0	09/30/2003	17C0F1E86AEg				-551606.08	551606.08	6334691.9283	@@test	
0	09/30/2003	17C0F1E86AEg				-551606.08	551606.08	6334691.9283	@@test	
0	12/31/2003	17C0F1E86AEh				373864.52	-373864.52	5960827.4083	@@test	
0	12/31/2003	17C0F1E86AEh				373864.52	-373864.52	5960827.4083	@@test	
0	06/30/2004	17C0F1E86AEi				1359881.752	-1359881.752	4600945.6563	@@test	
0	06/30/2004	17C0F1E86AEi				1359881.752	-1359881.752	4600945.6563	@@test	

#### After formatting

F7879E									
Date	Date	Contributions	Distributions	Current Value	Net Cashflow	Total Invested	Total Distributed	NAV Adjust	Description
06/30/03		(5,783,086)	0	0	(5,783,086)			5,783,086	f
09/30/03		(551,606)	0	0	(551,606)			551,606	g
12/31/03		0	373,865	0	373,865			(373,865)	h
06/30/04		0	1,359,882	0	1,359,882			(1,359,882)	i
12/31/07		0	1,108,000	0	1,108,000			(1,108,000)	j
09/30/08		0	1,159,000	0	1,159,000			(1,159,000)	k
06/30/09		0	1,165,497	0	1,165,497			(1,165,497)	l
06/30/10		0	2,039,035	0	2,039,035			(1,168,449)	m
03/31/12		0	0	11,000,000	0	(6,334,692)	7,205,278	11,000,000	n
01/01/03		(100)	0	0	(100)			100	a
03/31/12		0	200	0	200			(100)	b
06/30/12		(200)	0	0	(200)			200	

### 5.17.5.2. Features being illustrated

- PROC FORMAT step
- PICTURE statement

### 5.17.5.3. Program

//Create a personalized format. Low-high allows for different formatting on conditions of values: Positive, negative, and zero

PROC FORMAT;

PICTURE p\_money

LOW-HIGH ='#,##0; (#,##0); 0';

RUN;

//%PARAM IQID;

%PARAM PORT\_ID LABEL = "IQID:" TYPE=STRING INFORMAT="??"  
PICKID(VCPORTFOLIO;PORTFOLIO);

%PARAM ASATDATE LABEL = "To Date:" TYPE = DATE DEFAULT=TODAY; //  
NOTNULL;

```
%INCLUDE P_CF_BLOB_EXTRACTION;

DATA WORK.T_CF_DATA_CUBE_PORTFOLIO;

SET T_CF_BLOB_EXTRACTION (RENAME = (FUND=FUND_IQID) WHERE
%ASATDATE >= DATE);

RUN;

...

//In the following we only show the code that is related to the formatting. All the other
data processing code has been left out!

DATA T_CF_DATA_CUBE_PORTFOLIO (DROP = FUND_IQID COMPANY_IQID
PORTFOLIO PORTFOLIO_IQID CASH_IN CASH_OUT VALUATION NET_CASHFLOW
TOTALINVESTED

TOTALDISTRIBUTED NAV_ADJUST CUMULATIVE_NAV
MANAGEMENT_COMPANY_IQID INVESTMENT_IQID
ORIGINAL_DEAL_PARTNER_IQID CURRENT DEAL_PARTNER_IQID);

SET T_CF_DATA_CUBE_PORTFOLIO (WHERE PORTFOLIO_IQID IN (%PORT_ID));

//TO CONVERT VALUES INTO STRING TO GIVE NEGATIVES () INSTEAD OF - SIGN

COLUMN CASH_IN_CON TYPE=DOUBLEFORMAT=p_money.;

COLUMN CASH_OUT_CON TYPE=DOUBLEFORMAT=p_money.;

COLUMN VALUATION_CON TYPE=DOUBLEFORMAT=p_money.;

COLUMN NET_CASHFLOW_CON TYPE=DOUBLEFORMAT=p_money.;

COLUMN TOTALINVESTED_CON TYPE=DOUBLEFORMAT=p_money.;

COLUMN TOTALDISTRIBUTED_CON TYPE=DOUBLEFORMAT=p_money.;

COLUMN NAV_ADJUST_CON TYPE=DOUBLEFORMAT=p_money.;

COLUMN CASH_IN_CON = CASH_IN;
```

```
CASH_OUT_CON = CASH_OUT;  
VALUATION_CON = VALUATION;  
NET_CASHFLOW_CON = NET_CASHFLOW;  
TOTALINVESTED_CON = TOTALINVESTED;  
TOTALDISTRIBUTED_CON = TOTALDISTRIBUTED;  
NAV_ADJUST_CON = NAV_ADJUST;  
RUN;  
  
PROC PRINT DATA = T_CF_DATACUBE_PORTFOLIO;  
RUN;  
  
PROC SORT DATA = T_CF_DATACUBE_FUND;  
BY COMPANY DATE POSITION_DATE CURRENTCOST LASTVALUATION;  
RUN;
```

## 5.18. Examples - PROC GCHART step

The following examples illustrate the PROC GCHART step, and subordinate statements:

- [PROC GCHART - HBAR3D](#)

### 5.18.1. PROC GCHART - HBAR3D

#### 5.18.1.1. Goal

Generate a 3-dimensional horizontal bars' chart.

#### 5.18.1.2. Features being illustrated

- [PROC GCHART step](#)
- [HBar3D statement](#)

#### 5.18.1.3. Program

//Create the alias that points to the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Define the chart

```
PROC GCHART DATA=[CHART] SORTDESC STYLE=(CHARTCOLOR="lightgrey"  
CHARTEXECOLOR="darkgray" SERIESHAPE="DIAMOND"  
SERIEMULTICOLORS=True BARCOMBINEMETHOD="SIDE");
```

//Define the chart title

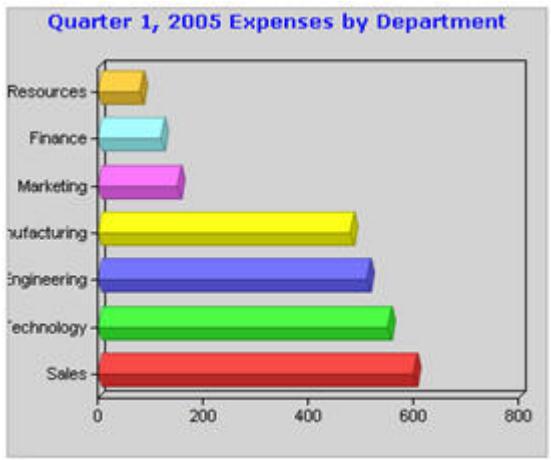
```
TITLE "Quarter 1, 2005 Expenses by Department" ( STYLE=(FOREGROUND="blue"  
FONT_FACE="Verdana" FONT_SIZE="10" FONT_WEIGHT="BOLD"));
```

//Define the chart type

```
HBAR3D DEPARTMENT(TYPE=MEAN SUMVAR=EXPENSES);
```

RUN;

## 5.18.1.4. Result



## 5.19. Examples - PROC GPLOT step

The following examples illustrate the PROC GPLOT step, and subordinate statements:

- [PROC GPLOT - PLOT](#)

### 5.19.1. PROC GPLOT - PLOT

#### 5.19.1.1. Goal

Generate a 3-dimensional horizontal bars' chart.

#### 5.19.1.2. Features being illustrated

- [PROC GCHART step](#)
- [HBAR3D statement](#)

#### 5.19.1.3. Program

//Create the alias that points to the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

//Define the plot chart

```
PROC GPLOT DATA=[CHART] STYLE=(WIDTH="560" CHARTCOLOR="lightgrey"  
CHARTEXECOLOR="darkgray" SERIESHAPE="DIAMOND"  
SERIEMULTICOLORS=True BARCOMBINEMETHOD="SIDE");
```

//Define the chart title

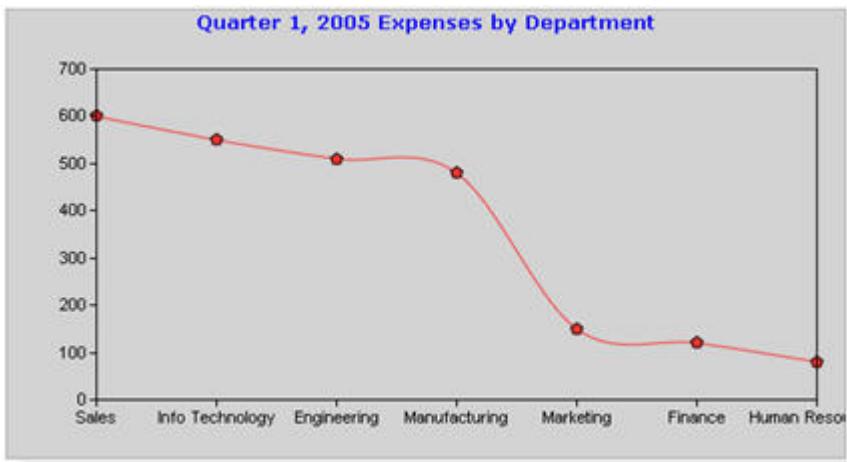
```
TITLE "Quarter 1, 2005 Expenses by Department" ( STYLE=(FOREGROUND="blue"  
FONT_FACE="Verdana" FONT_SIZE="10" FONT_WEIGHT="BOLD"));
```

//Define the chart type

```
PLOT EXPENSES*DEPARTMENT( INTERPOL=SPLINE SYMBOL=POLYGON);
```

RUN;

## 5.19.1.4. Result



## 5.20. Examples - PROC MEANS step

The following examples illustrate the PROC MEANS step, and subordinate statements:

- PROC MEANS - CLASS - N - LABEL
- PROC MEANS - JOIN
- PROC MEANS - JOIN - FIRST - LAST
- PROC MEANS - LEFT() - N
- PROC MEANS - MEAN - SUM - SQRT()
- PROC MEANS - MISSING
- PROC MEANS - PROC SORT - MAX
- PROC MEANS - SUM
- PROC MEANS - SUM - MEAN - N
- PROC MEANS - SUM - SEMESTER()
- PROC MEANS - IRR
- PROC MEANS - VAR - MULTIPLE

### 5.20.1. PROC MEANS - CLASS - N - LABEL

#### 5.20.1.1. Goal

Perform a simple statistical analysis.

#### 5.20.1.2. Features being illustrated

- PROC MEANS step
- CLASS statement
- N statement
- LABEL option

#### 5.20.1.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

```
/*Perform the statistical analysis -
```

```
Define input and output tables*/
```

```
PROC MEANS DATA = TEST.USERS OUT = WORK.USERS;  
//Select the column to perform the statistical analysis on  
CLASS CITY;  
//Launch the analysis  
N LASTNAME (LABEL = "Number of employees");  
RUN;  
//Display the table  
PROC PRINT DATA = WORK.USERS LABEL;  
RUN;
```

#### 5.20.1.4. Result

	City	Number of employees
1	New York	2
2	Paris	1
3	Bordeaux	2

### 5.20.2. PROC MEANS - JOIN

#### 5.20.2.1. Goal

Regroup a list of values corresponding to a single value.

#### 5.20.2.2. Features being illustrated

- [PROC MEANS step](#)
- [JOIN statement](#)

#### 5.20.2.3. Program

//Build an example data table

```
DATA WORK.t;  
COLUMN C;  
COLUMN X WIDTH=200;  
  
C = "Cat A"; X = "Value 1"; OUTPUT;  
C = "Cat A"; X = "Value 2"; OUTPUT;  
C = "Cat A"; X = "Value 3"; OUTPUT;  
C = "Cat B"; X = "Value 4"; OUTPUT;  
C = "Cat B"; X = "Value 5"; OUTPUT;  
  
RUN;  
  
//Print the example table  
  
PROC PRINT DATA=WORK.t;  
RUN;  
  
PROC MEANS DATA=WORK.t;  
CLASS C;  
JOIN X(DELIMITER=", ");  
RUN;  
  
PROC PRINT DATA=WORK.;  
RUN;
```

### 5.20.2.4. Result



	C	X
1	Cat A	Value 1
2	Cat A	Value 2
3	Cat A	Value 3
4	Cat B	Value 4
5	Cat B	Value 5

	C	X_JOIN
1	Cat A	Value 1, Value 2, Value 3
2	Cat B	Value 4, Value 5

### 5.20.3. PROC MEANS - JOIN - FIRST - LAST

#### 5.20.3.1. Goal

Regroup a list of values corresponding to a single value, and display first and last values.

#### 5.20.3.2. Features being illustrated

- PROC MEANS step
- JOIN statement
- FIRST statement
- LAST statement

#### 5.20.3.3. Program

//Build an example data table

```
DATA WORK.t;
COLUMN C;
COLUMN X WIDTH=200;
C = "Cat A"; X = "Value 1"; OUTPUT;
```

```
C = "Cat A"; X = "Value 2"; OUTPUT;  
C = "Cat A"; X = "Value 3"; OUTPUT;  
C = "Cat B"; X = "Value 4"; OUTPUT;  
C = "Cat B"; X = "Value 5"; OUTPUT;  
RUN;  
  
//Print the example table  
  
PROC PRINT DATA=WORK.t;  
RUN;  
  
PROC MEANS DATA=WORK.t;  
CLASS C;  
FIRST X;  
LAST X;  
JOIN X(DELIMITER=", ");  
RUN;  
  
PROC PRINT DATA=WORK.;  
RUN;
```

### 5.20.3.4. Result



The diagram illustrates the transformation of a source table into a result table using PROC MEANS. A red arrow points from the source table to the result table.

	C	X
1	Cat A	Value 1
2	Cat A	Value 2
3	Cat A	Value 3
4	Cat B	Value 4
5	Cat B	Value 5

	C	X_FIRST	X_LAST	X_JOIN
1	Cat A	Value 1	Value 3	Value 1, Value 2, Value 3
2	Cat B	Value 4	Value 5	Value 4, Value 5

## 5.20.4. PROC MEANS - LEFT() - N

### 5.20.4.1. Goal

For an existing table listing user data, identify the first letter of each name, and count the instances.

### 5.20.4.2. Features being illustrated

- PROC MEANS step
- Function operating on character strings

### 5.20.4.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

```
/*Prepare the table to analyze -
```

```
Copy the original table*/
```

```
DATA TEST.EX_USERS;  
SET TEST.USERS;  
  
//Create the a new column  
  
COLUMN CAR1_LASTNAME LABEL= "Lastname starts with";  
  
//Extract the first letter of each lastname  
  
CAR1_LASTNAME = LEFT(LASTNAME,1);  
  
RUN;  
  
//Perform the statistical analysis  
  
PROC MEANS DATA=TEST.EX_USERS;  
CLASS CAR1_LASTNAME;  
N LASTNAME (LABEL = "Total of letters");  
RUN;  
  
//Display the result  
  
PROC PRINT DATA = TEST.EX_USERS;  
RUN;
```

#### 5.20.4.4. Result

	CAR1_LASTNAME	LASTNAME
1	S	3
2	Z	1

#### 5.20.5. PROC MEANS - MEAN - SUM - SQRT()

##### 5.20.5.1. Goal

Calculate the correlation coefficient of two series X and Y.

### 5.20.5.2. Features being illustrated

- PROC MEANS step
- MEAN statement
- SUM statement
- Functions - numbers

### 5.20.5.3. Program

```
// Prepare the table with series X and Y
```

```
DATA WORK.SERIES;
```

```
COLUMN X TYPE = DOUBLE;
```

```
COLUMN YTYPE = DOUBLE;
```

```
X = 2;
```

```
Y = 3;
```

```
OUTPUT;
```

```
X = 4;
```

```
Y = 3;
```

```
OUTPUT;
```

```
X = 5;
```

```
Y = 2;
```

```
OUTPUT;
```

```
RUN;
```

```
//Calculate mean values for series X and Y
```

```
PROC MEANS DATA = WORK.SERIES OUT = WORK.MEAN;
```

```
MEAN X;  
MEAN Y;  
RUN;  
DATA WORK.MEAN;  
SET WORK.MEAN;  
%LET X_MEAN = X_MEAN;  
%LET Y_MEAN = Y_MEAN;  
RUN;  
//Calculate correlations  
DATA WORK.SERIES;  
SET WORK.SERIES;  
COLUMN SUP TYPE = DOUBLE;  
COLUMN INF_X TYPE = DOUBLE;  
COLUMN INF_Y TYPE = DOUBLE;  
SUP = (X - %X_MEAN) * (Y - %Y_MEAN);  
INF_X = (X - %X_MEAN) * (X - %X_MEAN);  
INF_Y = (Y - %Y_MEAN) * (Y - %Y_MEAN);  
RUN;  
PROC MEANS DATA = WORK.SERIES OUT = WORK.SERIES;  
SUM SUP (NAME = SUP);  
SUM INF_X (NAME = INF_X);
```

```
SUM INF_Y (NAME = INF_Y);  
RUN;  
  
DATA WORK.SERIES;  
  
SET WORK.SERIES;  
  
COLUMN INF TYPE = DOUBLE;  
  
COLUMN CORRELATION TYPE = DOUBLE;  
  
INF = SQRT(INF_X) * SQRT(INF_Y);  
  
CORRELATION = SUP / INF;  
  
RUN;
```

## 5.20.6. PROC MEANS - MISSING

### 5.20.6.1. Goal

For a group of values, calculate statistics, omitting or processing missing values.

### 5.20.6.2. Features being illustrated

- [PROC MEANS step](#)

### 5.20.6.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
  
DATA TEST.INITIAL_VALUES;  
  
COLUMN I TYPE=FLOAT;  
  
COLUMN J TYPE=FLOAT;  
  
I=1; J = NULL; OUTPUT;
```

```
I=2; J = 200; OUTPUT;  
I=3; J = 300; OUTPUT;  
I=4; J = 400; OUTPUT;  
I=5; J = 500; OUTPUT;  
I=6; J = NULL; OUTPUT;  
I=7; J = NULL; OUTPUT;  
RUN;  
  
PROC PRINT DATA=TEST.INITIAL_VALUES;  
RUN;  
  
PROC MEANS DATA=TEST.INITIAL_VALUES OUT=TEST.STATISTICS_1;  
N J;  
SUM J;  
MEAN J;  
FIRST J;  
LAST J;  
RUN;  
  
PROC PRINT DATA=TEST.STATISTICS_1;  
RUN;  
  
PROC MEANS DATA=TEST.INITIAL_VALUES OUT=TEST.STATISTICS_2 MISSING;  
N J;  
SUM J;
```

```
MEAN J;  
FIRST J;  
LAST J;  
RUN;  
PROC PRINT DATA=TEST.STATISTICS_2;  
RUN;
```

#### 5.20.6.4. Result

DATA=TEST.INITIAL\_VALUES

	I	J
1	1	
2	2	200
3	3	300
4	4	400
5	5	500
6	6	
7	7	

DATA=TEST.STATISTICS\_1

	J_N	J_SUM	J_FIRST	J_LAST	J_MEAN
1	4	1400	200	500	350

DATA=TEST.STATISTICS\_2

	J_N	J_SUM	J_FIRST	J_LAST	J_MEAN
1	7	1400			200

#### 5.20.7. PROC MEANS - CLASS - NOBS

##### 5.20.7.1. Goal

Count the number of rows in one aggregated category.

### 5.20.7.2. Features being illustrated

- PROC MEANS step
- CLASS statement
- NOBS statement

### 5.20.7.3. Program

```
DATA work.temp;  
  
COLUMN A TYPE=string;  
  
COLUMN B TYPE=double;  
  
A = "Cat1"; B=1; OUTPUT;  
  
A = "Cat1"; B=2; OUTPUT;  
  
A = "Cat1"; B=3; OUTPUT;  
  
A = "Cat2"; B=1; OUTPUT;  
  
A = "Cat2"; B=2; OUTPUT;  
  
RUN;  
  
PROCMEANS DATA=work.temp;  
  
CLASS A;  
  
NOBS number_of_observations;  
  
RUN;  
  
PROC PRINTDATA=work.temp;  
  
RUN;
```

## 5.20.7.4. Result

	A	NUMBER_OF_OBSERVATIONS
1	Cat1	3
2	Cat2	2
3	Cat3	1

## 5.20.8. PROC MEANS - PROC SORT - MAX

### 5.20.8.1. Goal

Make sure that the data line containing the MAX column value is the first line of the data table.

### 5.20.8.2. Features being illustrated

- PROC SORT step
- PROC MEANS step
- MAX statement

### 5.20.8.3. Program

//Sort the input table according to the column to which to apply the MAX statement

```
PROC SORT DATA=[WORK].[FUNDOP_DISTRIB] ;
```

```
BY FUND_OPERATION_ID FUND_ID INVESTOR_ID OPERATION_ID DESCENDING  
REFERENCE_DATE;
```

```
RUN;
```

//Apply the MAX statement, and retrieve the line being associated

```
PROC MEANS DATA=[WORK].[FUNDOP_DISTRIB] OUT=[WORK].  
[FUNDOP_DISTRIB_LASTDATE];
```

```
MAX REFERENCE_DATE(NAME=LASTDATE_DISTRIB LABEL="Last Distribution  
date");
```

```
CLASS FUND_ID INVESTOR_ID;
```

```
VAR CASH FUND_LABEL;  
RUN;
```

#### 5.20.8.4. Comment

- When using the MAX statement (or any other statement that returns one single value) within a PROC MEANS step, the line where the MAX value has been retrieved, can be any line of the data table. Therefore you cannot be sure that the VAR value corresponds to the line where the MAX value has been retrieved. To avoid mismatch, you have to make sure that the line containing the MAX value is the first line of the data table. To achieve this, you have to precede the PROC MEANS step by a PROC SORT step, and sort the data table according to the column to which you want to return the MAX value (descending sort).

#### 5.20.9. PROC MEANS - SUM

##### 5.20.9.1. Goal

For an existing table listing contracts, unit prices, and quantities order, calculate the total amount of contracts.

##### 5.20.9.2. Features being illustrated

- COLUMN statement
- PROC MEANS step
- SUM statement

##### 5.20.9.3. Program

```
//Create the alias that points the existing table location
```

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

```
//Copy the existing table to the WORK library
```

```
DATA WORK.AMOUNT_CONTRACTS;
```

```
SET TEST.CONTRACTS;
```

```
//Add a new column
```

```
COLUMN AMOUNT TYPE = DOUBLE LABEL = "QTY by unit price";  
//Calculate the values of the new column  
AMOUNT = QTY*UNIT_PRICE;  
RUN;  
//Calculate the statistics  
PROC MEANS DATA=WORK.AMOUNT_CONTRACTS.  
SUM AMOUNT (NAME = TOTAL_CONTRACTS LABEL="Total contracts");  
RUN;  
//Display the result  
PROC PRINT DATA = AMOUNT_CONTRACTS;  
RUN;
```

#### 5.20.9.4. Result

	TOTAL_CON
1	82062,4

### 5.20.10. PROC MEANS - SUM - MEAN - N

#### 5.20.10.1. Goal

Use two existing tables, one listing users and one listing contracts, to build a statistics table that shows for each user, the number of contracts, the amount being spent, and the average quantity of items ordered.

#### 5.20.10.2. Features being illustrated

- PROC MEANS step
- SUM statement

- [MEAN statement](#)
- [N statement](#)

### 5.20.10.3. Program

```
//Create the alias that points the existing table location  
  
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
  
//Prepare the table to analyze  
  
//Drop irrelevant columns  
  
DATA TEST.USERS_INFO_ABOUT_CONTRACTS (DROP = BIRTHDAY ZIP CITY);  
  
SET TEST.USERS_AND_CONTRACTS (WHERE UNIT_PRICE IS NOT NULL);  
  
RUN;  
  
//Calculate the statistics  
  
PROC MEANS DATA=TEST.USERS_INFO_ABOUT_CONTRACTS  
OUT=WORK.RESULTS;  
CLASS USER_ID;  
N USER_ID (NAME = N_CONTRACTS);  
SUM UNIT_PRICE (NAME = AMOUNT);  
MEAN QTY (NAME = AVERAGE);  
RUN;  
  
//Display the result  
  
PROC PRINT DATA = TEST.CONTRACT_PER_USER;  
RUN;
```

## 5.20.10.4. Result

	USER_ID	N_CONTRAC	AMOUNT	AVERAGE
1	101	4	19450,3	2,75
2	102	2	6441,6	50,5
3	103	1	7501	3

## 5.20.11. PROC MEANS - SUM - SEMESTER()

### 5.20.11.1. Goal

We start out with the following table:

Transaction date	Investment	Instrument	Transaction type	Amount in Euros
05/01/2006	Luna Atlantic	Bond A	BUY	325000
08/03/2006	Azur	Bond A	BUY	120000
05/04/2006	Trocadero	Bond A	BUY	725000
08/05/2006	Luna Atlantic	Bond A	SELL	-30000
08/06/2006	Belfast	Bond A	BUY	340000
08/07/2006	AZW	Bond A	BUY	480000

We want to regroup transactions by semester, and calculate statistics, such as total amount and average investment, per semester.

### 5.20.11.2. Features being illustrated

- PROC MEANS step
- CLASS statement
- SEMESTER() function
- N statement

### 5.20.11.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\{Private\}\TEST\EXAMPLES\TablesFormation";
```

//Select the values to perform the calculation on

```

PROC MEANS DATA = TEST.INPUT_FUND_TRANSACTIONS
OUT =TEST.OUTPUT_FUND_TRANSACTIONS;
CLASS SEMESTER(TRANSACTION_DATE);
SUM AMOUNT (NAME = TOTAL_AMOUNTS LABEL= "Total amounts");
MEAN AMOUNT (NAME = AVERAGE_INVESTMENT LABEL= "Average investment");
N TRANSACTION_DATE (NAME = TOTAL_TRANSACTIONS LABEL= "Total of
transactions");
RUN;
//Display the result
PROC PRINT DATA = TEST.OUTPUT_FUND_TRANSACTIONS LABEL NOOBS;
RUN;

```

#### 5.20.11.4. Comment

- Notice that in CLASS SEMESTER(TRANSACTION\_DATE) the value returned for TRANSACTION\_DATE is the **last day of the semester**, and not the number of the semester within the year, value that would be returned when SEMESTER(<date>) is being anywhere else than in a CLASS statement. Ex.:
 

```

PROC PRINT;
PUT SEMESTER ("19/08/2007");
RUN;
Result= 2
      
```
- The comment above applies as well to the following functions: YEAR(<date>), MONTH(<date>), QUARTER(<date>).

#### 5.20.11.5. Result

Transaction date	Total of transactions	Total amounts	Average investment
30/06/2006	5	1480000	296000
31/12/2006	1	480000	480000

## 5.20.12. PROC MEANS - IRR

### 5.20.12.1. Goal

We want to calculate the IRR for a fund that lists the following fund operations:

Operation date	Operation type	Total called	Total distributed	Valuation
02/01/2002	Closing	95000	0	0
10/05/2003	Distribution	0	79000	0
18/12/2003	Late closing & call	1500	0	0
15/03/2004	Valuation	0	0	200000

To be able to use the IRR statement properly, it is important that the table reflects real fund conditions, i.e. the first operation after the **Closing** must be a **negative flux**.

### 5.20.12.2. Features being illustrated

- PROC MEANS step
- IRR statement

### 5.20.12.3. Program

// Prepare the table for IRR calculation

```
DATA TEST.INPUT_FUND_OPERATIONS;
SET TEST.INPUT_FUND_OPERATIONS;
COLUMN AMOUNT TYPE = FLOAT;
// Group amount on one column...
AMOUNT = -TOTAL_CALLED + TOTAL_DISTRIBUTED + VALUATION;
RUN;
//Print the intermediary table
```

```

PROC PRINT DATA=TEST.INPUT_FUND_OPERATIONS NOOBS LABEL;
RUN;

//Calculate the IRR

PROC MEANS DATA=TEST.INPUT_FUND_OPERATIONS OUT=TEST.OUTPUT_IRR;
IRR AMOUNT (NAME = IRR_NET LABEL = "Net IRR{F}TRI Net"
DATE=OPERATION_DATE);

RUN;

//Print the resulting table

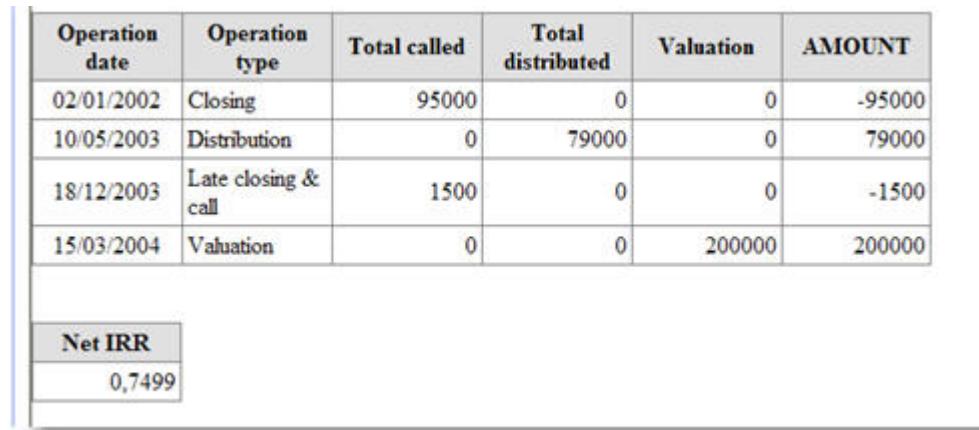
PROC PRINT DATA=TEST.OUTPUT_IRR LABEL NOOBS;
RUN;

```

#### 5.20.12.4. Comment

- Notice that an intermediary table has to be built. The intermediary table adds an **AMOUNT** column to the initial table. The **AMOUNT** column calculates cashflow using as input the **TOTAL\_CALLED**, **TOTAL\_DISTRIBUTED**, and **VALUATION** columns.
- The IRR statement can then be executed on the **AMOUNT** column.

#### 5.20.12.5. Result



The screenshot shows a software interface with a table of fund operations and a calculated Net IRR.

Operation date	Operation type	Total called	Total distributed	Valuation	AMOUNT
02/01/2002	Closing	95000	0	0	-95000
10/05/2003	Distribution	0	79000	0	79000
18/12/2003	Late closing & call	1500	0	0	-1500
15/03/2004	Valuation	0	0	200000	200000

Below the table, a box displays the calculated Net IRR:

**Net IRR**  
0,7499

## 5.20.13. PROC MEANS - VAR - MULTIPLE

### 5.20.13.1. Goal

For a data table, that lists total values and cost, calculate a multiple.

### 5.20.13.2. Features being illustrated

- PROC MEANS step
- VAR statement

### 5.20.13.3. Program

//Create an alias

LIBNAME user ".";

//Create and instantiate test data table

DATA TT;

COLUMN NAME TYPE=STRING;

COLUMN TOTAL\_VALUE TYPE=FLOAT;

COLUMN COST TYPE=FLOAT;

NAME="A"; TOTAL\_VALUE=50; COST=100;OUTPUT;

NAME="B"; TOTAL\_VALUE=150; COST=120;OUTPUT;

NAME="C"; TOTAL\_VALUE=-170; COST=-150;OUTPUT;

RUN;

//Calculate multiple without residual value

PROC MEANS DATA=TT OUT=TT2;

MULTIPLE TOTAL\_VALUE(LABEL="Multiple");

```
RUN;  
  
//Display data on screen  
  
PROC PRINT DATA=TT2;  
  
TITLE "Without RESIDUALVALUE";  
  
RUN;  
  
//Calculate multiple with residual value  
  
PROC MEANS DATA=TT OUT=TT2;  
  
MULTIPLE TOTAL_VALUE(RESIDUALVALUE=COST LABEL="Multiple");  
  
RUN;  
  
//Display data on screen  
  
PROC PRINT DATA=TT2;  
  
TITLE "With RESIDUALVALUE";  
  
RUN;
```

#### 5.20.13.4. Comment

- To calculate a multiple, you need a negative value among the data!

## 5.21. Examples - PROC OFFICE step

The following examples illustrate the PROC OFFICE step, subordinate statements and controls:

- [PROC OFFICE - Generate XML report](#)
- [PROC OFFICE - Generate multi WORD document generator](#)

### 5.21.1. PROC OFFICE - Generate XML report

#### 5.21.1.1. Goal

Generate an XML report using eFront Invest datamart data tables.

#### 5.21.1.2. Features being illustrated

- [PROC OFFICE step](#)
- [TEMPLATE statement](#)
- [eFront Invest XML Schema Definition](#)

#### 5.21.1.3. Program

//Generate an XML report

LIBNAME USER ":";

PROC OFFICE FILE ="Export.xml";

TEMPLATE "&TestTemplate.xml";

SET Funds Companies Securities;

RUN;

### 5.21.1.4. Template XML file used to produce the XML report



```
<?xml version="1.0" encoding="UTF-8"?>
<Test>
    <ef_attrib name = "creationDateTime"><ef_macro name = "Report_Date" format = "y
    <ef_loop table = "Funds">
        <Fund>
            <ef_attrib name = "Name"><ef_value column = "Fund_Name"/></ef_attrib>
            <ef_loop table = "Companies" where="Investor_ID=?Fund_ID">
                <PortfolioCompany>
                    <Name><ef_value column = "Company_Name"/></Name>
                    <ef_loop table = "Securities" where="Company_ID=?Company_ID">
                        <Instrument>
                            <Name><ef_select expr = "Security_Name">
                                <ef_case value = "">Unnamed security</ef_case>
                                <ef_otherwise>
                                    <ef_value column = "Security_Name"/>
                                </ef_otherwise>
                            </ef_select>
                        </Name>
                        <ef_select expr = "Security_Type_Code">
                            <ef_case value = "1">
                                <ef_attrib name = "Type">Loan</ef_attrib>
                            </ef_case>
                            <ef_case value = "2">
                                <ef_attrib name = "Type">Shares</ef_attrib>
                                <NBShares><ef_value column = "NB_Shares" format = "###,###.00"/></NBShares>
                            </ef_case>
                            <ef_otherwise>
                                <ef_attrib name = "Type">Other</ef_attrib>
                            </ef_otherwise>
                        </ef_select>
                        <CurrentValuation><ef_value column = "Adjusted_Valuation"/>
                    </Instrument>
                </ef_loop>
            </PortfolioCompany>
        </ef_loop>
        <FundSize><ef_value column = "Fund_Size" format = "##,##.00"/></FundSize>
    </Fund>
</ef_loop>
</Test>
```

## 5.21.1.5. Tables being used to feed the XML report

### Funds

FUND_NAME	FUND_ID	FUND_SIZE
Fund Tamino IV	123DDDE32	34200000
Fund Papageno	429DFDE35	1593400.17
Fund Euridice	153DAAE37	4669030.0234

### Companies

INVESTOR_ID	COMPANY_NAME	COMPANY_ID
429DFDE35	Brothers & Brothers	43EE5CA17
123DDDE32	Henry Co.	35A3B17C1
429DFDE35	EcoChemicals Inc.	45F79CD42
153DAAE37	Banana Corporation	78A34CD56
153DAAE37	Ananas Republic	AEF5C432

### Securities

SECURITY_ID	SECURITY_NAME	SECURITY_TYPE_CODE	NB SHARES	ADJUSTED_VALUATION	COMPANY_ID
123	LOAN		1	340299	43EE5CA17
234			2	4280	345323.33 43EE5CA17
345	Some other instrument		5	333	200000 35A3B17C1
567	SHARES		2	50040	500400 45F79CD42
678	LOAN		1	450300	45F79CD42
456	Shares		2	435	435000.17 45F79CD42
789	WARRANTS		4	1	320003.44 AEF5C432
890	Other		5	453	66600 78A34CD56

## 5.21.1.6. Result

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Test creationDateTime="2009-Jun-01">
- <Fund Name="Fund Tamino IV">
- <PortfolioCompany>
  <Name>Henry Co.</Name>
  - <Instrument Type="Other">
    <Name>Some other instrument</Name>
    <CurrentValuation>200,000.00</CurrentValuation>
  </Instrument>
</PortfolioCompany>
<FundSize>34,200,000.00</FundSize>
</Fund>
- <Fund Name="Fund Papageno">
- <PortfolioCompany>
  <Name>Brothers & Brothers</Name>
  - <Instrument Type="Loan">
    <Name>LOAN</Name>
    <CurrentValuation>340,299.00</CurrentValuation>
  </Instrument>
  - <Instrument Type="Shares">
    <Name>Unnamed security</Name>
    <NBShares>4,280</NBShares>
    <CurrentValuation>345,323.33</CurrentValuation>
  </Instrument>
</PortfolioCompany>
- <PortfolioCompany>
  <Name>EcoChemicals Inc.</Name>
  - <Instrument Type="Shares">
    <Name>SHARES</Name>
    <NBShares>50,040</NBShares>
    <CurrentValuation>500,400.00</CurrentValuation>
  </Instrument>
  - <Instrument Type="Loan">
    <Name>LOAN</Name>
    <CurrentValuation>450,300.00</CurrentValuation>
  </Instrument>
  - <Instrument Type="Shares">
    <Name>Shares</Name>
    <NBShares>435</NBShares>
    <CurrentValuation>435,000.17</CurrentValuation>
  </Instrument>
</PortfolioCompany>
<FundSize>1,593,400.17</FundSize>
</Fund>
- <Fund Name="Fund Euridice">
```

## 5.21.2. PROC OFFICE - Generate muti WORD document generator

### 5.21.2.1. Goal

Generate a multi WORD document generator.

### 5.21.2.2. Features being illustrated

- PROC OFFICE step

### 5.21.2.3. Program

```
// Count the number of documents you want to generate

DATA WORK.T_PARTNERS_INFO;

SET WORK.T_PARTNERS_INFO (WHERE INVESTMENT_ID IN
(%SUBSCRIBERS_ID));

COLUMN LINE_COUNTER TYPE=INTEGER;

LINE_COUNTER = _N_;

%NB_LETTER = _N_;

RUN;

// for as many document you need to generate...

%FOR %COUNTER = 1 %TO %NB_LETTER

DATA LETTER_OUTPUT.T_PARTNERS_INFO_PER_SUBSCRIBER (DROP=BUFFER
BUFFER_NAME BUFFER_NAME2 BUFFER_RATE BUFFER_FORMAT
CAPITAL_CALL COMMITMENT REMAINING_COMMITMENT DISTRIBUTION);

SET WORK.T_PARTNERS_INFO (where LINE_COUNTER=%COUNTER);

RUN;

// Make your document for each line

PROC OFFICE FILE=%FILE_NAME DIRECTORY="\\YourSharedNetworkFolder" ;

SET LETTER_OUTPUT.T_PARTNERS_INFO_PER_SUBSCRIBER;

TEMPLATE "&R3 - Template EN v2.0.doc";

RUN;
```

%Next;

## 5.22. Examples - PROC PRINT step

The following examples illustrate the PROC PRINT step, and subordinate statements:

- Example - IRR CASH FLOWS BY INSTRUMENT
- PROC PRINT - BY - COLLAPSE - EXPAND
- PROC PRINT - EVENT PREDATAROW
- PROC PRINT - LABEL
- PROC PRINT - NOOBS - N
- PROC PRINT - N - FORMAT - SUM - BY
- PROC PRINT - PUT
- PROC PRINT - PUT-- Investor Summary (Report + Chart)
- PROC PRINT - SUM - STYLE
- PROC PRINT - SUM - BY - STYLE
- PROC PRINT - VAR - .SUM - STYLE
- PROC PRINT - VAR - .MULTIPLE

### 5.22.1. Example - IRR CASH FLOWS BY INSTRUMENT

#### 5.22.1.1. Goal

Produce the following report:

**Company RD Cards**

[Edit](#) [Reports](#) [Close](#)

**Main Info**

Region: eFront (Shared)

Company: RD Cards Status: Existing Borrower

**Select filters** [Execute](#)

For the period ending  [Search](#)

**FR Page Sample**

Sample [View](#)

IRR CASH FLOWS BY INSTRUMENT					
INSTRUMENT	INVESTOR	Transaction Date	TTYPE	Amount Due	Amount Paid
INSTRUMENT=Mezzanine					
Subtotal INSTRUMENT=Mezzanine				56,635.70	-15,000,000.00
INSTRUMENT=Second Lien					
Second Lien	DAM Invest	08/02/2007	Advance debt		-8,000,000.00
Second Lien	DAM Invest	08/02/2007	Advance debt		-2,500,000.00
Second Lien	DAM Invest	08/02/2007	Advance debt		-2,500,000.00
Second Lien	DAM Invest	08/02/2007	Purchase PK (Capitalised)	56,635.70	
Second Lien	DAM Invest	14/03/2007	Advance debt		-2,000,000.00
Subtotal INSTRUMENT=Second Lien				56,635.70	-15,000,000.00
INSTRUMENT=Senior TrB					
Subtotal INSTRUMENT=Senior TrB				56,635.70	-15,000,000.00
INSTRUMENT=Senior TrC					
Subtotal INSTRUMENT=Senior TrC				56,635.70	-15,000,000.00

### 5.22.1.2. Features being illustrated

- %INCLUDE statement
- %PARAM statement
- DATA step
- PROC PRINT step
- FREEZEHEADER option
- FREEZELEFT option
- NOGRANDTOTAL option
- URL option

### 5.22.1.3. Program

```
LIBNAME USER ":";  
  
//Link to stylesheet  
  
%INCLUDE "FV_FOR_PAGES";  
  
//Declare macro variables  
  
%PARAM ASAT LABEL="For the period ending" TYPE=DATE;  
  
%PARAM IQID LABEL="Company" TYPE=STRINGINFORMAT="??ID(SFAACOUNT)";  
  
//Create data table  
  
DATA WORK.T_FV_PAGES_SAMPLE (WHERE BORROWERID=%IQID);  
  
SET V_FV_PAGES_SAMPLE;  
  
RUN;  
  
//Display the result  
  
PROC PRINT DATA=WORK.T_FV_PAGES_SAMPLE LABEL N="" NOOBS  
URL=(TYPE="TXT_REPORT.FOLDER=F_VCINVESTINS" VAR=INSTRUMENT)  
FREEZEHEADERNOGRANDTOTALFREEZELEFT=1;  
  
VAR  
  
INSTRUMENT(STYLE(COLHEADER)=(WIDTH="200"))  
  
INVESTOR  
  
DATE(LABEL="Transaction Date")  
  
TTYPE  
  
.SUM(VAR=DUE FORMAT="#,##00"LABEL="Amount Due")  
  
.SUM(VAR=PAID FORMAT="#,##00"LABEL="Amount Paid");
```

```
TITLE = "IRR CASH FLOWS BY INSTRUMENT";
```

```
BY INSTRUMENT;
```

```
RUN;
```

## 5.22.2. PROC PRINT - BY - COLLAPSE - EXPAND

### 5.22.2.1. Goal

Print a table, where some groups are expanded, others are collapsed.

### 5.22.2.2. Features being illustrated

- PROC PRINT step
- BY statement
- VAR statement

### 5.22.2.3. Program

```
PROC PRINT DATA=Work.Table_By_Year FreezeHeader FreezeLeft = 1 NoGrandTotal;
```

```
ByYear(Expand= -%Nb_Years)Quarter(Collapse = 3);
```

```
//Expand: only the last %Nb_Year(s) will be expanded (all others are collapsed)
```

```
//Collapse: for the opened year(s), the first 3 quarters will be collapsed
```

```
VAR .Max(Var =Date) .Sum(Var =Value);
```

```
RUN;
```

### 5.22.2.4. Result

	DATE	VALUE
Year=2008		
	28-déc.-2008	615,78
Year=2009		
	28-déc.-2009	8 446,82
Year=2010		
Quarter=1		
	30-mars-2010	2 451,55
Quarter=2		
	30-juin-2010	2 604,73
Quarter=3		
	30-sept.-2010	2 767,48
Quarter=4		
23	30-oct.-2010	960,40
24	30-nov.-2010	980,00
25	31-déc.-2010	1 000,00
	31-déc.-2010	2 940,40
	31-déc.-2010	10 764,16

### 5.22.2.5. Comment

- Functions used within the VAR statement in a PROC PRINT step must be preceded by a period!

## 5.22.3. PROC PRINT - LABEL

### 5.22.3.1. Goal

Display all the column titles of a table using the variables' label.

### 5.22.3.2. Features being illustrated

- PROC PRINT step
- LABEL option

### 5.22.3.3. Program

//Display data on screen - column titles = variable names

PROC PRINT DATA=TEST.CONTRACTS;

RUN;

//Display data on screen - column titles = variable labels

PROC PRINT DATA=TEST.CONTRACTS LABEL;

RUN;

#### 5.22.3.4. Result

	CONTRACT	USER_ID	UNIT_PRICE	QTY
1	593	101	4220,1	2
2	594	101	1299	3
3	595	101	7501	1
4	596	102	11,4	100

	Contract Id	User Id	Unit Price	Quantity
1	593	101	4220,1	2
2	594	101	1299	3
3	595	101	7501	1
4	596	102	11,4	100

#### 5.22.4. PROC PRINT - NOOBS - N

##### 5.22.4.1. Goal

Print a table, count the total of lines, and display the total.

##### 5.22.4.2. Features being illustrated

- PROC PRINT step
- NOOBS option
- N option

##### 5.22.4.3. Program

//Display data on screen

PROC PRINT DATA=TEST.CONTRACTS NOOBS N="Total number of contracts : ";

RUN;

## 5.22.4.4. Result

CONTRACT	USER_ID	UNIT_PRICE	QTY
593	101	4220,1	2
593	101	1299	3
595	101	7501	1
596	102	11,4	100
597	102	6430,2	1
596	103	7501	3
597	101	6430,2	5
598	105	20030,2	5
Total number of contracts : 8			

## 5.22.5. PROC PRINT - N - FORMAT - SUM - BY

### 5.22.5.1. Goal

Print a table, count the sub-total of occurrences per group, and display totals.

### 5.22.5.2. Features being illustrated

- PROC PRINT step
- N option
- LABEL option

### 5.22.5.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

//Create a personalized format

PROC FORMAT;

PICTURE P\_Money

LOW-HIGH = "\$ 00.00";

RUN;

//Display data on screen

## PROC PRINT DATA=TEST.CONTRACTS LABEL NOOBS

```
N="Total number of contracts per customer: "
"Total number of contracts: ";
FORMAT UNIT_PRICE P_MONEY.;
SUM UNIT_PRICE(STYLE=(BACKGROUND=PINK));
BY USER_ID (STYLE=(FONT_FACE="Verdana"
FONT_SIZE="10pt" PADDING="10pt"));
RUN;
```

### 5.22.5.4. Result

Customer=101			
Contract Id	Customer	Amount	Items purchased
593	101	\$4220,10	2
594	101	\$1299,00	3
595	101	\$7501,00	1
599	101	\$6430,20	5
		\$19450,30	
Total number of contracts per customer: 4			

(The example listing doesn't explicitly list all customers!)

Customer=105			
Contract Id	Customer	Amount	Items purchased
600	105	\$20030,20	5
		\$20030,20	
		\$53423,10	
Total number of contracts per customer: 1			
Total number of contracts: 8			

## 5.22.6. PROC PRINT - PUT

### 5.22.6.1. Goal

Display the value of a macro-variable

### 5.22.6.2. Features being illustrated

- PROC PRINT step
- PUT statement
- %LET statement

### 5.22.6.3. Program

//Create a macro variable

```
%LET MY_STRING = "My string";
```

//Print the macro variable

```
PROC PRINT;
```

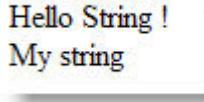
```
PUT "Hello String !";
```

```
PUT CRLF();
```

```
PUT %MY_STRING;
```

```
RUN;
```

### 5.22.6.4. Result



```
Hello String !
My string
```

## 5.22.7. PROC PRINT - PUT-- Investor Summary (Report + Chart)

### 5.22.7.1. Goal

Produce the following report:



### 5.22.7.2. Features being illustrated

- PROC PRINT step
- PUT statement

### 5.22.7.3. Program

```

LIBNAME USER ".";
LIBNAME COMMON "\{SHARED}\COMMON";
//Link to stylesheet
%INCLUDE "FV_FOR_PAGES";
//Declare macro variables
%PARAM ASAT LABEL="For the period ending" TYPE=DATE;
%PARAM IQID LABEL="Company" TYPE=STRINGINFORMAT="??ID(SFAACOUNT)";
//Link to program to manage currency rates
%INCLUDE "\{SHARED}\COMMON\P_FV_EXRATES";
//Create data table
DATA WORK.T_INVESTMENTS (WHERE BORROWERID=%IQID) ;

```

```
MERGE WORK.T_FV_EXRATES V_INVESTMENTS;  
BY CCY;  
COLUMN OUTSTANDINGEUR TYPE=DOUBLE;  
IF EXRATE IS NULL THEN EXRATE=1;  
END;  
OUTSTANDINGEUR =ROUND(OUTSTANDING/EXRATE,2);  
IF STATUS IS NOT NULLTHEN OUTPUT;  
END;  
RUN;  
//Sort data  
PROC SORT DATA=WORK.T_INVESTMENTS (KEEP= INVESTOR INSTRUMENT  
OUTSTANDING OUTSTANDINGEUR )  
OUT= WORK.T_INVESTMENTS;  
BY INSTRUMENT INVESTOR;  
RUN;  
//Calculate statistics  
PROC MEANS DATA=WORK.T_INVESTMENTS  
OUT=WORK.T_CHART_INVESTMENTS;  
CLASS INVESTOR;  
SUM OUTSTANDING (NAME=OUTSTANDING LABEL="Principal");  
SUM OUTSTANDINGEUR (NAME=OUTSTANDINGEUR LABEL="Principal (EUR)");  
RUN;
```

```
//Style page to be printed as html output: Open the first cell of an html table
PROC PRINT;
PUT "table<TR><TD>";
RUN;

//Generate chart and write it in the first table cell

PROC GCHART DATA=WORK.T_CHART_INVESTMENTS
SORTDESC STYLE=(WIDTH=300 CHARTCOLOR="lightgrey"
CHARTEXECOLOR="darkgray" SERIESHAPE="DIAMOND"
SERIEMULTICOLORS=True BARCOMBINEMETHOD="SIDE");
TITLE "Investments (EUR) by ultimate Investor" (STYLE=(FOREGROUND="BLUE"
FONT_FACE="VERDANA"
FONT_SIZE="10" FONT_WEIGHT="BOLD"));
HBAR3D INVESTOR (TYPE=SUM SUMVAR=OUTSTANDINGEUR);
RUN;

//Close the first cell, and open the second cell of an html table
PROC PRINT;
PUT "</TD><TD>";
RUN;

//Print report and write it in the second table cell

PROC PRINT DATA=WORK.T_FV_PAGES_SAMPLE LABEL N="" NOOBS
URL=(TYPE="TXT_REPORT.FOLDER=F_VCINVESTINS" VAR=INSTRUMENT)
FREEZEHEADERNOGRANDTOTALFREEZELEFT=1;
VAR
INVESTOR (STYLE(COLHEADER)=(WIDTH="200") Label="Investor")
```

```
DATE(LABEL="Transaction Date")
TTYPE
.SUM(VAR=OUTSTANDING FORMAT="#,##0.00"LABEL="Principal")
.SUM(VAR=OUTSTANDINGEUR FORMAT="#,##00.00"LABEL="Principal (EUR)");
TITLE = "List of Investors";
RUN;
//Close second cell of the html table + close line + table
PROC PRINT;
PUT "</TD></TR></TABLE>";
RUN;
```

## 5.22.8. PROC PRINT - SUM - STYLE

### 5.22.8.1. Goal

Print a table listing information about customers and items, and print the total of unit prices.

### 5.22.8.2. Features being illustrated

- PROC PRINT step
- SUM statement
- STYLE option

### 5.22.8.3. Program

```
//Create an alias
```

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

```
//Create a personalized format
```

```
PROC FORMAT;
```

## PICTURE P\_Money

```
LOW-HIGH = "$ 00.00";  
RUN;  
  
//Display data on screen  
  
PROC PRINT DATA=TEST.CONTRACTS LABEL;  
  
FORMAT UNIT_PRICE P_MONEY.;  
  
SUM UNIT_PRICE(STYLE=(BACKGROUND=PINK));  
RUN;
```

### 5.22.8.4. Result

	Contract Id	User Id	Unit Price	Quantity
1	593	101	\$4220,10	2
2	594	101	\$1299,00	3
3	595	101	\$7501,00	1
4	596	102	\$11,40	100
5	597	102	\$6430,20	1
6	596	103	\$7501,00	3
7	597	101	\$6430,20	5
8	598	105	\$20030,20	5
			\$53423,10	

### 5.22.9. PROC PRINT - SUM - BY - STYLE

#### 5.22.9.1. Goal

Print a table listing information about customers and items. Calculate the total of unit prices and display per contract.

#### 5.22.9.2. Features being illustrated

- PROC PRINT step
- SUM statement
- BY statement

- **STYLE option**

### 5.22.9.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a personalized format

```
PROC FORMAT;
```

```
PICTURE P_Money
```

```
LOW-HIGH = "$ 00.00";
```

```
RUN;
```

//Display data on screen

```
PROC PRINT DATA=TEST.CONTRACTS LABEL N="";
```

```
FORMAT UNIT_PRICE P_MONEY.;
```

```
SUM UNIT_PRICE(STYLE=(BACKGROUND=PINK));
```

```
BY CONTRACT_ID (STYLE=(FONT_FACE="Verdana"
```

```
FONT_SIZE="10pt" PADDING="10pt"));
```

```
RUN;
```

### 5.22.9.4. Result

**Contract Id=593**

	Contract Id	User Id	Unit Price	Quantity
1	593	101	\$4220,10	2
2	593	101	\$1299,00	3
			<b>\$5519,10</b>	

**Contract Id=595**

	Contract Id	User Id	Unit Price	Quantity
3	595	101	\$7501,00	1
			<b>\$7501,00</b>	

## 5.22.10. PROC PRINT - URL

### 5.22.10.1. Goal

Define a link that allows to access variable values.

### 5.22.10.2. Features being illustrated

- PROC PRINT step
- URL option

### 5.22.10.3. Program

```
PROC PRINT DATA=WORK.T_INVESTMENTS LABEL N="Number of Investors"  
URL=(TYPE="TXT_REPORT.FOLDER=F_VCINVESTINS" VAR=INSTRUMENT)  
FREEZEHEADER NOGRANDTOTAL FREEZELEFT=1 NOOBS;
```

VAR

```
INSTRUMENT (STYLE(COLHEADER)=(WIDTH="200") LABEL="Instrument")  
  
INVESTOR (STYLE(COLHEADER)=(WIDTH="200") LABEL="Investor")  
  
.SUM(VAR=OUTSTANDING FORMAT="#,##0.00" LABEL="Principal")  
  
.SUM(VAR=OUTSTANDINGEUR FORMAT="#,##0.00" LABEL="Principal (EUR)");  
  
BY INSTRUMENT;  
  
TITLE "INVESTMENTS BY INSTRUMENTS";  
  
RUN;
```

### 5.22.10.4. Comment

1. Notice the URL option indicates that there is a clickable link on the column Instrument which supplies the instrument to the folder F\_VCINVESTINS therefore expanding to show the instrument details. If you view the source you will see the link is something like:

```
<tr class=row_even0 style="cursor:hand;"  
onclick="parent.ajaxrep_submitForm('TXT_REPORT.FOLDER=F_VCINVESTINS$Av  
iagen II TrB');">
```

It seems this calls a function in the Ajax interface which opens up the sub rows.

2. The .SUM clauses in the PROC PRINT handle the sub-totals of OUTSTANDING and OUTSTANDINGEUR respectively. The page shows the summary by instrument until you click the link when the totals by investor are shown.

## 5.22.11. PROC PRINT - VAR - .SUM - STYLE

### 5.22.11.1. Goal

From an existing table, print a selection of variables and apply a function to them

### 5.22.11.2. Features being illustrated

- PROC PRINT step
- VAR statement

### 5.22.11.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

//Create a personalized format

```
PROC FORMAT;
```

```
PICTURE P_Money
```

```
LOW-HIGH = "$ 00.00";
```

```
RUN;
```

//Display data on screen

```
PROC PRINT DATA=TEST.CONTRACTS_SORTED LABEL;
```

```
VAR
```

```

USER_ID(LABEL="Customer" CONTRACT_ID
.SUM(VAR = UNIT_PRICE FORMAT = P_MONEY.
STYLE(TOTAL) = (FONT_FACE = "Arial" FONT_SIZE = "14pt"));
RUN;

```

#### 5.22.11.4. Result

	Customer	Contract Id	Amount
1	101	593	\$4220,10
2	101	594	\$1299,00
3	101	595	\$7501,00
4	101	599	\$6430,20
5	102	596	\$11,40
6	102	597	\$6430,20
7	103	598	\$7501,00
8	105	600	\$20030,20
			<b>\$53423,10</b>

#### 5.22.11.5. Comment

- Functions used within the VAR statement in a PROC PRINT step must be preceded by a period!

### 5.22.12. PROC PRINT - VAR - .MULTIPLE

#### 5.22.12.1. Goal

For a data table, that lists total values and cost, calculate a multiple.

#### 5.22.12.2. Features being illustrated

- PROC PRINT step
- VAR statement

#### 5.22.12.3. Program

//Create an alias

```
LIBNAME user ".";
```

```
//Create and instantiate test data table  
  
DATA TT;  
  
COLUMN NAME TYPE=STRING;  
  
COLUMN TOTAL_VALUE TYPE=FLOAT;  
  
COLUMN COST TYPE=FLOAT;  
  
NAME="A"; TOTAL_VALUE=50; COST=100;OUTPUT;  
  
NAME="B"; TOTAL_VALUE=150; COST=120;OUTPUT;  
  
NAME="C"; TOTAL_VALUE=-170; COST=-150;OUTPUT;  
  
RUN;  
  
//Calculate multiple with residual value and display data on screen  
  
PROC PRINT DATA=TT;  
  
NAME  
  
COST  
  
TOTAL_VALUE  
  
.MULTIPLE(LABEL="MULTIPLE" VAR=TOTAL_VALUE RESIDUALVALUE=COST);  
  
RUN;
```

#### 5.22.12.4. Comment

- Functions used within the VAR statement in a PROC PRINT step must be preceded by a period!
- To calculate a multiple, you need a negative value among the data!

## 5.23. Examples - PROC PRINTCOL step

The following examples illustrate the PROC PRINTCOL step, and subordinate statements:

- [PROC PRINTCOL - FLOW - STYLESHEET](#)

### 5.23.1. PROC PRINTCOL - FLOW - STYLESHEET

#### 5.23.1.1. Goal

Write a column of a data table into 3 columns to be printed.

#### 5.23.1.2. Features being illustrated

- [PROC PRINTCOL step](#)
- [STYLESHEET step](#)

#### 5.23.1.3. Program

//Create an alias

```
LIBNAME TEST "\\\EFRONT_SUP.2(Private)\TEST\EXAMPLES\Tables";
```

```
LIBNAME VIEWS "\\\EFRONT_SUP.2(Private)\TEST\EXAMPLES\Views";
```

//Set up styles for the printed table

```
STYLESHEET;
```

```
PRINTCOL(DATA)=(WIDTH="250px" BORDER="NONE" FONT_SIZE="8pt"  
FONT_FACE="TAHOMA");
```

```
STYLESHEET END;
```

//Extract data from the Efront data base

```
DATA TEST.COMPANIES;
```

```
SET VIEWS.Company_FundOperation;
```

```
RUN;  
  
// Sort the data table and limit lines  
  
PROC SORT DATA = TEST.COMPANIES TOP=15 NODUPKEY;  
BY COMPANY;  
  
RUN;  
  
//Create the table to print, option: vertical flow  
  
PROC PRINTCOL DATA=TEST.COMPANIES FLOW=VERTICALLY COLUMNS=3  
STYLE(TITLE)= (BACKGROUND="GRAY");  
  
VAR COMPANY;  
  
TITLE "List of companies: vertical flow";  
  
RUN;  
  
//Create the table to print, option: horizontal flow  
  
PROC PRINTCOL DATA=TEST.COMPANIES FLOW=HORIZONTALLY COLUMNS=3  
STYLE(TITLE)= (BACKGROUND="RED");  
  
VAR COMPANY;  
  
TITLE "List of companies: horizontal flow";  
  
RUN;
```

## 5.23.1.4. Result

List of companies: vertical flow		
21 CENTRALE PARTNERS	ABN AMRO - NSM ENTREPRISES	ADVENTION BUSINESS PARTNERS
4 T - Traduction & Interprétaire	ABN AMRO CAPITAL FRANCE	AFG
ABELIA CONSULTING	ACLAND	AFIC
ABN AMRO - BANQUE DE NEUILIZE	ACTIVA CAPITAL	AFORGE
	ADVENT	AGF

List of companies: horizontal flow		
21 CENTRALE PARTNERS	4 T - Traduction & Interprétaire	
ABELIA CONSULTING	ABN AMRO - NSM ENTREPRISES	
ABN AMRO CAPITAL FRANCE	ACLAND	ACTIVA CAPITAL
ADVENT	ADVENTION BUSINESS PARTNERS	AFG
AFIC	AFORGE	AGF

## 5.24. Examples - PROC PRINTFORM step

The following examples illustrate the PROC PRINTFORM step, and subordinate statements:

- [PROC PRINTFORM - TITLE - VAR](#)
- [PROC PRINTFORM - TABLENOBS\(\) - %IF %THEN %END](#)

### 5.24.1. PROC PRINTFORM - TITLE - VAR

#### 5.24.1.1. Goal

Write the first column of a data table in form 'style'.

#### 5.24.1.2. Features being illustrated

- [PROC FORMAT step](#)
- [TITLE statement](#)
- [VAR statement](#)

#### 5.24.1.3. Program

/Write the first column of a data table in form 'style'

```
PROC PRINTFORM DATA=T_PROJECT_LIST LABEL;  
  TITLE "Deal Info";  
  VAR  
    DEAL_NAME (LABEL="Deal Name")  
    COMPANY (LABEL="Company"  
             URL=(TYPE="TXT_REPORT.FOLDER=F_SFAACCOUNT" VAR=COMPANY_IQID))  
    FUND (LABEL="DS Fund")  
    STAGE (LABEL="Deal Stage")  
    JNLDATE2 (LABEL="Date recent stage change")
```

DATE\_RECEIVED (LABEL="Date deal originally logged")

BUS\_DES (LABEL="Business Description")

ADD\_ON\_CODE (LABEL="Platform/Add-on")

COUNTRY (LABEL="Main Country of Operation")

DEALTYPE (LABEL="Deal type")

ESTIMATED\_EV (FORMAT="#,##" LABEL="Estimated EV (€m)")

ESTIMATED\_TE (FORMAT="#,##" LABEL="Estimated Total Equity (€m)")

DS\_SHAREHOLDING (FORMAT="0" LABEL="DS Shareholding (%))

SECTOR

SUB\_SECTOR;

RUN;

#### 5.24.1.4. Result

Deal Info	
Deal Name	Test
Company	
DS Fund	
Deal Stage	Exited
Date recent stage change	02/12/2008 5:27:58 PM
Date deal originally logged	19/06/2008
Business Description	
Platform/Add-on	
Main Country of Operation	United Kingdom
Deal type	ACQ
Estimated EV (€m)	
Estimated Total Equity (€m)	
DS Shareholding (%)	
Sector	
Sub sector	

## 5.24.2. PROC PRINTFORM - TABLENOBS() - %IF %THEN %END

### 5.24.2.1. Goal

According to the number of rows in a table, print/don't print the table.

### 5.24.2.2. Features being illustrated

- PROC PRINTFORM step
- Functions - special
- %IF...%THEN...%END

### 5.24.2.3. Program

```
%IF TABLENOBS ("WORK.T_PROJECTC_LIST_REJECTED") <> 0 %THEN
```

```
PROC PRINTFORM DATA WORK.T_PROJECT_LIST_REJECTED LABEL;
```

```
TITLE "Rejection";
```

```
%END;
```

### 5.24.2.4. COMMENT

According to the value returned by TABLENOBS(), the table is being printed or not.

## 5.25. Examples - PROC SENDTOIC step

The following examples illustrate the PROC SENDTOIC step, and subordinate statements:

- [PROC SENDTOIC](#)

### 5.25.1. PROC SENDTOIC

#### 5.25.1.1. Goal

Send a call notice from eFront Invest to Investment Café Classic using an eFront Script report.

#### 5.25.1.2. Features being illustrated

- [PROC SENDTOIC step](#)

#### 5.25.1.3. Program

//assign values to variable

```
%LET FilePath = '\Shared region(company)\CallNotice.docx';  
%LET Investor = "0D020AA255D74AC598DDF3DA0A5C7C90";  
%LET Fund = "B5176D8F61B74F5C881F400E616FE0F2";  
%LET Batch = null;  
//send document from FI to IC  
  
PROC SENDTOIC  
  
FILE = %FilePath  
  
TITLE = "Call-February"  
  
FUND = %Fund  
  
TYPE = "CALL"
```

INVESTOR = %Investor;

NOTICENAME = "First version of call notice"

DATE = Now();

BATCH = %Batch;

RUN;

## 5.26. Examples - PROC SORT step

The following examples illustrate the PROC SORT step, and subordinate statements:

- [PROC SORT - NODUPKEY - BY](#)
- [PROC SORT - BY - UPCASE](#)

### 5.26.1. PROC SORT - NODUPKEY - BY

#### 5.26.1.1. Goal

Sort an existing table by alphabetical order.

#### 5.26.1.2. Features being illustrated

- [PROC SORT step](#)
- [BY statement](#)

#### 5.26.1.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

//Launch the sorting procedure on the existing table

```
PROC SORT DATA = TEST.USERS
```

```
OUT = TEST.USERS_SORTED
```

```
NODUPKEY;
```

```
BY LASTNAME;
```

```
RUN;
```

//Display data on screen

```
PROC PRINT DATA = TEST.Users_SORTED;
```

```
RUN;
```

## 5.26.1.4. Result

	USER_ID	FIRSTNAME	LASTNAME	BIRTHDAY	ZIP	CITY
1	101	Bobby	Schmidt	05/03/1959	10021	New York
2	103	Sandra	Specker	01/12/1962	33000	Bordeaux
3	102	Ruber	Zodi	01/25/1980	75001	Pans

## 5.26.2. PROC SORT - BY - UPCASE

### 5.26.2.1. Goal

Sort an existing table by alphabetical order. We start out with the following table:

Transaction date	Investment	Instrument	Transaction type	Amount in Euros
05/01/2006	Luna Atlantic	Bond A	BUY	325000
08/03/2006	Azur	Bond A	BUY	120000
05/04/2006	Trocadero	Bond A	BUY	725000
08/05/2006	luna Atlantic	Bond A	SELL	-30000
08/06/2006	belfast	Bond A	BUY	340000
08/07/2006	aZW	Bond A	BUY	480000

Our objective is to sort the table according to the investment. As you can see Investment labels start alternatively with an upcase or a lowercase letter.

### 5.26.2.2. Features being illustrated

- PROC SORT step
- BY statement

### 5.26.2.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "<Private>\TEST\EXAMPLES\TablesFormation";
```

```
/*Sort the transaction table without using the UPCASE keyword*/
```

```
PROC SORT DATA = TEST.INPUT_FUND_TRANSACTIONS
```

```
OUT = TEST.FUND_TRANSACTIONS_SORTED  
NODUPKEY;  
BY INVESTMENT;  
RUN;  
//Display data on screen  
PROC PRINT DATA=TEST.FUND_TRANSACTIONS_SORTED NOOBS LABEL;  
RUN;  
//Sort the transaction table using the UPCASE keyword  
PROC SORT DATA = TEST.INPUT_FUND_TRANSACTIONS  
OUT = TEST.FUND_TRANSACTIONS_SORTED  
NODUPKEY;  
BY UPCASE INVESTMENT;  
RUN;  
//Display data on screen  
PROC PRINT DATA=TEST.FUND_TRANSACTIONS_SORTED NOOBS LABEL;  
RUN;
```

#### 5.26.2.4. Result

Using By without Upcase:

Transaction date	Investment	Instrument	Transaction type	Amount in Euros
08/03/2006	Azur	Bond A	BUY	120000
05/01/2006	Luna Atlantic	Bond A	BUY	325000
05/04/2006	Trocadero	Bond A	BUY	725000
08/07/2006	aZW	Bond A	BUY	480000
08/06/2006	belfast	Bond A	BUY	340000
08/05/2006	luna Atlantic	Bond A	SELL	-30000

Using By with Upcase:

Transaction date	Investment	Instrument	Transaction type	Amount in Euros
08/03/2006	Azur	Bond A	BUY	120000
08/07/2006	aZW	Bond A	BUY	480000
08/06/2006	belfast	Bond A	BUY	340000
05/01/2006	Luna Atlantic	Bond A	BUY	325000
05/04/2006	Trocadero	Bond A	BUY	725000

## 5.27. Examples - PROC SQLIMPORT step

The following examples illustrate the PROC SQLIMPORT step:

- [PROC SQLIMPORT - TABLE - FILTER](#)
- [PROC SQLIMPORT - CONNECTION - SQL - MAXROWS](#)
- [PROC SQLIMPORT - SQL - MAXROWS - ??FILTER](#)
- [PROC SQLIMPORT - VIEW](#)
- [PROC SQLIMPORT - COLUMNMAPPING](#)

### 5.27.1. PROC SQLIMPORT - TABLE - FILTER

#### 5.27.1.1. Goal

Import data from a database related to the application into an eFront Report table.

#### 5.27.1.2. Features being illustrated

- [PROC SQLIMPORT](#)

#### 5.27.1.3. Program

//Import data from the underlying database into an eFront Report table.

```
PROC SQLIMPORT DATA=WORK.T_FUNDS TABLE="VCFUND" FILTER;
```

```
RUN;
```

#### 5.27.1.4. Comment

- Imports the data of the database table **VCFUND** into the eFront Report table **T\_Funds**.
- Data are filtered according to the user regions.

### 5.27.2. PROC SQLIMPORT - CONNECTION - SQL - MAXROWS

#### 5.27.2.1. Goal

Import data from a database that is different from the underlying application database into an eFront Report table.

### 5.27.2.2. Features being illustrated

- PROC SQLIMPORT

### 5.27.2.3. Program

//Import data from a database that is different from the underlying application database into an eFront Report table.

```
PROC SQLIMPORT DATA=WORK.T_FUNDS CONNECTION="OtherDataSource"  
SQL="select * from VCFUND" MAXROWS=100;
```

RUN;

### 5.27.2.4. Comment

- "OtherDataSource" refers to the name of the connection that allows the program to access the external database; defined in the file **efront.config**.
- Imports up to 100 rows of the database table **VCFUND** into the eFront Report table **T\_Funds**.

## 5.27.3. PROC SQLIMPORT - SQL - MAXROWS - ??FILTER

### 5.27.3.1. Goal

Import data from the underlying application database into an eFront Report table, select the number of rows to import and filter according to user regions.

### 5.27.3.2. Features being illustrated

- PROC SQLIMPORT

### 5.27.3.3. Program

//Import data from a database that is different from the underlying application database into an eFront Report table.

```
PROC SQLIMPORT DATA=WORK.T_FUNDS SQL="select * from VCFUND A  
WHERE ??FILTER" MAXROWS=100;
```

RUN;

### 5.27.3.4. Comment

- Imports up to 100 rows of the database table **VCFUND** into the eFront Report table **T\_Funds**.
- Filters data according to user regions.

## 5.27.4. PROC SQLIMPORT - VIEW

### 5.27.4.1. Goal

Import data from the underlying application database into an eFront Report table, using a view to specify the data to import.

### 5.27.4.2. Features being illustrated

- [PROC SQLIMPORT](#)

### 5.27.4.3. Program

//Import data from the underlying application database into an eFront Report table.

```
PROC SQLIMPORT DATA=WORK.T_FUNDS VIEW=LIBRARY_VIEW.FV_V_FUND;  
RUN;
```

### 5.27.4.4. Comment

You can replace the program above by:

```
PROC DATA=WORK.T_FUNDS;  
SET LIBRARY_VIEW.FV_V_FUND;  
RUN;
```

But, the SQLIMPORT step is faster!

## 5.27.5. PROC SQLIMPORT - COLUMNMAPPING

### 5.27.5.1. Goal

Import a table from a database and map one of its columns to a column in a eFront Report mapping table. Insert an illustration of the result if possible. Note that this

example uses a table from the eFront Data Warehouse, but the features illustrated can be used on tables from any connected database.

### 5.27.5.2. Features being illustrated

- [PROC SQLIMPORT step](#)
- COLUMNMAPPING option

### 5.27.5.3. Program

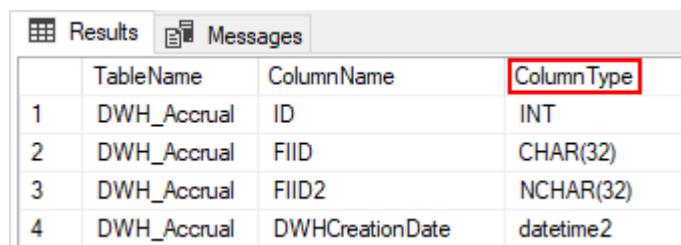
```
// Create a table with the mapping.  
  
DATA WORK.MAPPING;  
  
COLUMN COLDB;  
  
COLUMN COLFS;  
  
COLDB="ColumnType";  
  
COLFS="Type";  
  
OUTPUT;  
  
RUN;  
  
// Import the database table and create the column mapping.  
  
PROC SQLIMPORT  
DATA=WORK.IMPORT  
TABLE="DWH_Columns"  
CONNECTION ="DWH"  
COLUMNMAPPING=WORK.MAPPING  
COLUMNMAPPINGDBNAME="COLDB"  
COLUMNMAPPINGFSNAME="COLFS";
```

RUN;

```
// Print the imported table  
PROC PRINT DATA=WORK.IMPORT;  
  
RUN;
```

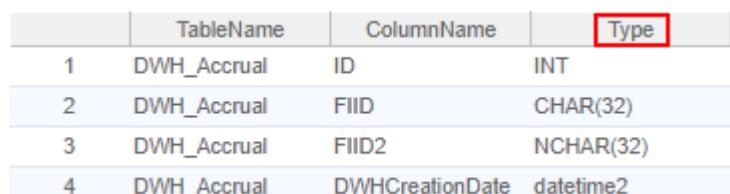
#### 5.27.5.4. Result

- Source eFront Data Warehouse table



	TableName	ColumnName	ColumnType
1	DWH_Accrual	ID	INT
2	DWH_Accrual	FIID	CHAR(32)
3	DWH_Accrual	FIID2	NCHAR(32)
4	DWH_Accrual	DWHAccrualDate	datetime2

- Imported eFront Data Warehouse table



	TableName	ColumnName	Type
1	DWH_Accrual	ID	INT
2	DWH_Accrual	FIID	CHAR(32)
3	DWH_Accrual	FIID2	NCHAR(32)
4	DWH_Accrual	DWHAccrualDate	datetime2

#### 5.27.5.5. Comment

- CONNECTION="DWH" allows the program to access the eFront Data Warehouse. When performing actions on the eFront Invest database, the CONNECTION option does not need to be specified.

## 5.28. Examples - PROC SQLTABLE step

The following examples illustrate the PROC SQLTABLE step:

- [PROC SQLTABLE - 1](#)
- [PROC SQLTABLE - 2](#)
- [PROC SQLTABLE - COLUMNMAPPING](#)

### 5.28.1. PROC SQLTABLE - (1)

#### 5.28.1.1. Goal

Export an eFront Report table to an SQL database.

#### 5.28.1.2. Features being illustrated

- [PROC SQLTABLE step](#)

#### 5.28.1.3. Program

//Export an eFront Report table to a SQL database.

```
PROC SQLTABLE  
  CONNECTION="MyConnection" DATA=FV_T_PORTCO_TRANSACTIONS  
  TABLE="DATAMART_PORTCO_TRANSACTIONS" CREATE NOERROR;
```

RUN;

#### 5.28.1.4. Comment

- "MyConnection" refers to the name of the connection that allows the program to access the external database.

### 5.28.2. PROC SQLTABLE - (2)

#### 5.28.2.1. Goal

Export an eFront Report table to an SQL database.

### 5.28.2.2. Features being illustrated

- PROC SQLTABLE step
- SQL statement

### 5.28.2.3. Program

//Build a test table.

```
DATA Work.TestTable;  
COLUMN Id;  
COLUMN Name;  
COLUMN MyTime Type = DATE;  
Id = "E3CB55";  
Name = "Toto";  
MyTime = Now();  
OUTPUT;
```

RUN;  
//Create an SQL table in the external database and export the content of the table  
Work.TestTable into it

```
PROC SQLTABLE CONNECTION="External_DB" GROUPBY=500  
DATA=Work.TestTable TABLE="Test_Data" CREATE NOERROR;
```

RUN;

//Test the content of the exported data table and re-import the table

```
DATA Work.TestTable_2;  
SQL "Select Name, Mytime from Test_Data" Connection = "External_DB";
```

COLUMN Name;

COLUMN Time Type = "DATE";

RUN;

//Print the imported table

PROC PRINT DATA = Work.TestTable\_2;

RUN;

#### 5.28.2.4. Result

Execute		
	NAME	TIME
1	Toto	25/07/2019 11:29...

#### 5.28.2.5. Extract of eFront.config

```
efront.config:  
  <db>  
    <ConnectionString type="string">Data  
    Source=ApplicationDB...</ConnectionString>  
    <GenerateIndexesSeparately  
    type="bool">false</GenerateIndexesSeparately>  
    <br>  
    </br>  
    <NamedConnections>  
      <Connection1>  
        <Name type="string">External_DB</Name>  
        <ConnectionString type="string">Data  
    Source=SomeDB...</ConnectionString>  
        </Connection1>  
      </NamedConnections>  
  </db>
```

#### 5.28.2.6. Comment

- "External\_DB" refers to the name of the connection that allows the program to access the external database.

## 5.28.3. PROC SQLTABLE - COLUMNMAPPING

### 5.28.3.1. Goal

Export a eFront Report table to a database and map a column in the eFront Report table to a column in the exported database table. Note that this example uses a table from the eFront Data Warehouse, but the features illustrated can be used on tables from any connected database.

### 5.28.3.2. Features being illustrated

- [PROC SQLTABLE step](#)
- COLUMNMAPPING option

### 5.28.3.3. Program

//Create a eFront Report table to export to the database.

DATA WORK.FRUIT;

COLUMN Id;

COIUMN Fruit;

Id=125;

Name="Apple";

OUTPUT;

Id=126;

Name="Peach";

OUTPUT;

Id=127;

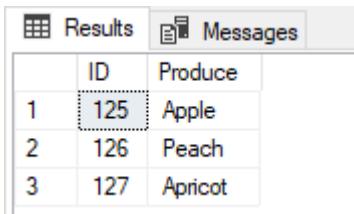
Name="Apricot";

OUTPUT;

```
RUN;  
  
// Create a table with the mapping.  
  
DATA WORK.MAPPING;  
  
COLUMN COLDB;  
  
COLUMN COLFS;  
  
COLDB="Produce";  
  
COLFS="Fruit";  
  
OUTPUT;  
  
RUN;  
  
// Export the eFront Report table to the database and create the column mapping.  
  
PROC SQLTABLE  
  
DATA=WORK.FRUIT  
  
TABLE="DWH_PRODUCE"  
  
CONNECTION ="DWH"  
  
CREATE  
  
NOERROR  
  
COLUMNMAPPING=WORK.MAPPING  
  
COLUMNMAPPINGDBNAME="COLDB"  
  
COLUMNMAPPINGFSNAME="COLFS";  
  
RUN;
```

### 5.28.3.4. Result

From Select \* FROM DWH\_Produce



The screenshot shows a software interface with two tabs at the top: "Results" (selected) and "Messages". Below the tabs is a table with three rows of data:

ID	Produce
1	125 Apple
2	126 Peach
3	127 Apricot

### 5.28.3.5. Comment

- CONNECTION="DWH" allows the program to access the eFront Data Warehouse. When performing actions on the eFront Invest database, the CONNECTION option does not need to be specified.

## 5.29. Examples - PROC TABULATE step

The following examples illustrate the PROC TABULATE step, and subordinate statements:

- [PROC TABULATE - BOX - STYLE](#)
- [PROC TABULATE - TABLE - ALL - N - SUM](#)
- [PROC TABULATE - TABLE - STYLE \(1\)](#)
- [PROC TABULATE - TABLE - STYLE \(2\)](#)
- [PROC TABULATE - TITLE](#)

### 5.29.1. PROC TABULATE - BOX - STYLE

#### 5.29.1.1. Goal

Style the box above rows in a statistics table.

#### 5.29.1.2. Features being illustrated

- PROC TABULATE step
- BOX statement
- STYLE option

#### 5.29.1.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

//Define the statistical data table

```
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;
```

```
CLASS USER_ID;
```

```
VAR UNIT_PRICE;
```

```
TABLE (USER_ID ALL="Total"), UNIT_PRICE*(N SUM);
```

//Style the box above rows

BOX (LABEL = 'Customers and Purchased Items'

STYLE = (BACKGROUND = WHITE BORDER = NONE

FONT\_FACE = "Verdana" FONT\_SIZE = '16pt'));

RUN;

#### 5.29.1.4. Result

Customers and Purchased Items	UNIT_PRICE	
	N	SUM
USER_ID	101	4 19450,3
	102	2 6441,6
	103	1 7501
	Total	1 20030,2
		8 53423,1

#### 5.29.2. PROC TABULATE - TABLE - ALL - N - SUM

##### 5.29.2.1. Goal

For a group of customers, calculate statistical information about items being purchased. Summarize the values of items being purchased and the amount being spent.

##### 5.29.2.2. Features being illustrated

- PROC TABULATE step
- TABLE statement
- ALL variable

##### 5.29.2.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

//Define the statistical data table

```
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;
```

```

CLASS USER_ID;
VAR UNIT_PRICE;
TABLE USER_ID = "Customer" ALL = "Total",
UNIT_PRICE = "Items"*
(N = "Number of Items" SUM = "Total Amount");
RUN;

```

#### 5.29.2.4. Result

		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
Total		8	53423,1

#### 5.29.3. PROC TABULATE - TABLE - STYLE (1)

Apply styles to elements of a eFront Script table.

##### 5.29.3.1. Features being illustrated

- PROC TABULATE step
- TABLE statement
- STYLE option

##### 5.29.3.2. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

and apply styles to particular items\*/

```
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;  
TITLE1 "Customer and Items";  
CLASS USER_ID;  
VAR UNIT_PRICE;  
TABLE USER_ID = "Customer" ALL=(LABEL="Total")  
STYLE(HEADER) = (BACKGROUND = YELLOW FOREGROUND = BLACK)  
STYLE(DATA) = (BACKGROUND = BLACK FOREGROUND = WHITE)),  
UNIT_PRICE = "Items"*(  
(N = "Number of Items" SUM = "Total Amount");  
RUN;
```

### 5.29.3.3. Result

		Customer and Items	
		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
Total		8	53423,1

### 5.29.4. PROC TABULATE - TABLE - STYLE (2)

#### 5.29.4.1. Goal

Apply styles to values returned by functions used in a TABLE statement.

## 5.29.4.2. Features being illustrated

- PROC TABULATE step
- TABLE statement
- STYLE option

## 5.29.4.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\\\EFRONT_SUP.2(Private)\\TEST\\EXAMPLES\\TablesFormation";
```

```
/*Define the statistical data table using the PROC TABULATE step
```

```
and apply styles to particular items*/
```

```
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;
```

```
TITLE1 "Customer and Items";
```

```
CLASS USER_ID;
```

```
VAR UNIT_PRICE;
```

```
TABLE USER_ID = "Customer" ALL=(LABEL="Total")
```

```
UNIT_PRICE = "Items"*
```

```
(N = (LABEL = "Number of Items" STYLE(DATA) = (BACKGROUND = PINK  
FOREGROUND = PURPLE))
```

```
SUM = "Total Amount")
```

```
;
```

```
RUN;
```

## 5.29.4.4. Result

		Customer and Items	
		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
Total		8	53423,1

## 5.29.5. PROC TABULATE - TITLE

### 5.29.5.1. Goal

Create a table title.

### 5.29.5.2. Features being illustrated

- PROC TABULATE step
- TITLE statement

### 5.29.5.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private)\TEST\EXAMPLES\TablesFormation";
```

and apply styles to particular items\*/

```
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;
```

```
TITLE1 "Customers and Purchased Items";
```

```
CLASS USER_ID;
```

```
VAR UNIT_PRICE;
```

```
TABLE USER_ID = "Customer" ALL = "Total",
UNIT_PRICE = "Items"*
(N = "Number of Items" SUM = "Total Amount")
;
RUN;
```

#### 5.29.5.4. Result

		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
	Total	8	53423,1

## 5.30. Examples - PROC TRANSPOSE step

The following examples illustrate the PROC TRANSPOSE step, and subordinate statements:

- [PROC TRANSPOSE \(1\)](#)
- [PROC TRANSPOSE \(2\)](#)
- [PROC TRANSPOSE - SingleCurrency Transactions](#)

### 5.30.1. PROC TRANSPOSE (1)

#### 5.30.1.1. Goal

Transpose the columns of a table into lines, and create a new output table.

#### 5.30.1.2. Features being illustrated

- [PROC TRANSPOSE step](#)

#### 5.30.1.3. Program

//Create an alias

```
LIBNAME TEST "\\\\EFRONT_SUP.2(Private)\\TEST\\EXAMPLES\\TablesFormation";
```

//Print data table

```
PROC PRINT DATA=TEST.FinancialData;
```

```
RUN;
```

//Transpose the columns of the data table into lines

```
PROC TRANSPOSE DATA=TEST.FinancialData  
OUT=TEST.FinancialData_TRANSPOSED;
```

```
RUN;
```

//Print the resulting table

```
PROC PRINT DATA=TEST.FinancialData_TRANSPOSED;
```

RUN;

### 5.30.1.4. Result

	YEAR	REVENUE	EBIT	EBITDA	HEADCOUNT
1	2004	100	1.5	1.2	200
2	2005	200	2.5	2.2	400
3	2006	300	3.5	3.2	500

	_NAME_	COL1	COL2	COL3
1	YEAR	2004	2005	2006
2	REVENUE	100	200	300
3	EBIT	1.5	2.5	3.5
4	EBITDA	1.2	2.2	3.2
5	HEADCOUNT	200	400	500

### 5.30.1.5. Comment

- The number of columns of the output table = the number of lines of the input table + 1.
- By default **eFront Script** generates automatically column names.

## 5.30.2. PROC TRANSPOSE (2)

### 5.30.2.1. Goal

Transpose the columns of a table into lines, and create a new output table.

### 5.30.2.2. Features being illustrated

- PROC TRANSPOSE step

### 5.30.2.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

//Print data table

```
PROC PRINT DATA=TEST.FinancialData;
```

```
RUN;
```

```
//Transpose the columns of the data table into lines
```

```
PROC TRANSPOSE DATA=TEST.FinancialData
OUT=TEST.FinancialData_TRANSPOSED NAME=DATA LABEL="Data";
```

```
VAR REVENUE EBIT EBITDA HEADCOUNT;
```

```
ID YEAR;
```

```
IDLABEL YEAR;
```

```
RUN;
```

```
//Print the resulting table
```

```
PROC PRINT DATA=TEST.FinancialData_TRANSPOSED LABEL;
```

```
RUN;
```

#### 5.30.2.4. Result

	YEAR	REVENUE	EBIT	EBITDA	HEADCOUNT
1	2004	100	1.5	1.2	200
2	2005	200	2.5	2.2	400
3	2006	300	3.5	3.2	500

	Data	2004	2005	2006
1	REVENUE	100	200	300
2	EBIT	1.5	2.5	3.5
3	EBITDA	1.2	2.2	3.2
4	HEADCOUNT	200	400	500

#### 5.30.2.5. Comment

- The number of columns of the output table = the number of lines of the input table + 1.
- The **ID** statement defines that the values of the input table column **YEAR** are the new column names. As **YEAR** values are numeric values, **eFront Script** prefixes column

names with an underscore. To get rid of the underscore in the display, you define **YEAR** also as being an **IDLABEL** column, and formatted values display as column titles without the underscore.

## 5.31. Examples - PROC TRANSPOSE2 step

The following examples illustrate the PROC TRANSPOSE2 step, and subordinate statements:

- [PROC TRANSPOSE2 - MultiCurrency Transactions](#)

To illustrate the difference between the usage of the steps PROC TRANSPOSE and PROC TRANSPOSE2, the example above has been adapted to the usage of PROC TRANSPOSE:

[PROC TRANSPOSE - SingleCurrency Transactions](#)

### 5.31.1. PROC TRANSPOSE2 - MultiCurrency Transactions

#### 5.31.1.1. Goal

Get the values of the initial table, transpose them into columns in the output table, and write selected values into these tables.

#### 5.31.1.2. Features being illustrated

- [PROC TRANSPOSE2 step](#)

#### 5.31.1.3. Program

```
LIBNAME USER ":";
```

```
//Build the table to be transposed
```

```
DATA WORK.TOTRANSPOSE;
```

```
COLUMN TRANSACTION TYPE=INTEGER;
```

```
COLUMN REFDATE TYPE=DATE;
```

```
COLUMN BASE_AMOUNT TYPE=FLOAT;
```

```
COLUMN BASE_CCY TYPE=STRING;
```

```
COLUMN CALC_CCY TYPE=STRING;
```

```
COLUMN CALC_AMOUNT TYPE=FLOAT;
```

```
TRANSACTION=1;REFDATE="17/04/2014";BASE_AMOUNT=10;BASE_CCY="EUR";C  
ALC_CCY="EUR";CALC_AMOUNT=10;OUTPUT;
```

```
TRANSACTION=1;REFDATE="17/04/2014";BASE_AMOUNT=10;BASE_CCY="EUR";C  
ALC_CCY="USD";CALC_AMOUNT=ROUND(13.558,2);OUTPUT;
```

```
TRANSACTION=1;REFDATE="17/04/2014";BASE_AMOUNT=10;BASE_CCY="EUR";C  
ALC_CCY="GBP";CALC_AMOUNT=ROUND(8.354916869,2);OUTPUT;
```

```
TRANSACTION=2;REFDATE="05/04/2014";BASE_AMOUNT=14;BASE_CCY="EUR";C  
ALC_CCY="EUR";CALC_AMOUNT=14;OUTPUT;
```

```
TRANSACTION=2;REFDATE="05/04/2014";BASE_AMOUNT=14;BASE_CCY="EUR";C  
ALC_CCY="USD";CALC_AMOUNT=ROUND(18.9812,2);OUTPUT;
```

```
TRANSACTION=2;REFDATE="05/04/2014";BASE_AMOUNT=14;BASE_CCY="EUR";C  
ALC_CCY="GBP";CALC_AMOUNT=ROUND(11.69688362,2);OUTPUT;
```

```
TRANSACTION=3;REFDATE="24/03/2014";BASE_AMOUNT=14;BASE_CCY="EUR";C  
ALC_CCY="EUR";CALC_AMOUNT=16;OUTPUT;
```

```
TRANSACTION=3;REFDATE="24/03/2014";BASE_AMOUNT=14;BASE_CCY="EUR";C  
ALC_CCY="USD";CALC_AMOUNT=ROUND(20.7808,2);OUTPUT;
```

```
TRANSACTION=3;REFDATE="24/03/2014";BASE_AMOUNT=14;BASE_CCY="EUR";C  
ALC_CCY="GBP";CALC_AMOUNT=ROUND(13.6432,2);OUTPUT;
```

```
TRANSACTION=4;REFDATE="18/03/2014";BASE_AMOUNT=10;BASE_CCY="EUR";C  
ALC_CCY="EUR";CALC_AMOUNT=10;OUTPUT;
```

```
TRANSACTION=4;REFDATE="18/03/2014";BASE_AMOUNT=10;BASE_CCY="EUR";C  
ALC_CCY="USD";CALC_AMOUNT=ROUND(13.558,2);OUTPUT;
```

```
TRANSACTION=4;REFDATE="18/03/2014";BASE_AMOUNT=10;BASE_CCY="EUR";C  
ALC_CCY="GBP";CALC_AMOUNT=ROUND(8.354916869,2);OUTPUT;
```

RUN;

//Print the table to be transposed

```
PROC PRINT DATA = WORK.TOTRANSPOSE;  
RUN;  
  
//Transpose the columns of the data table  
  
PROC TRANSPOSE2 DATA=WORK.TOTRANSPOSE  
OUT=WORK.TABLE_TRANSPOSED LABEL=CALC_CCY VALUE=CALC_AMOUNT;  
  
CLASS TRANSACTION;  
  
VAR REFDATE BASE_AMOUNT BASE_CCY;  
RUN;  
  
//Print the resulting table  
  
PROC PRINT DATA=WORK.TABLE_TRANSPOSED LABEL;  
RUN;
```

### 5.31.1.4. Result

#### PROC-TRANSPOSE2\_MultiCurrencyTransactions

Execute						
	TRANSACTION	REFDATE	BASE_AMOUNT	BASE_CCY	CALC_CCY	CALC_AMOUNT
1	1	17/04/2014	10 EUR	EUR		10
2	1	17/04/2014	10 EUR	USD		13,56
3	1	17/04/2014	10 EUR	GBP		8,35
4	2	05/04/2014	14 EUR	EUR		14
5	2	05/04/2014	14 EUR	USD		18,98
6	2	05/04/2014	14 EUR	GBP		11,7
7	3	24/03/2014	14 EUR	EUR		16
8	3	24/03/2014	14 EUR	USD		20,78
9	3	24/03/2014	14 EUR	GBP		13,64
10	4	18/03/2014	10 EUR	EUR		10
11	4	18/03/2014	10 EUR	USD		13,56
12	4	18/03/2014	10 EUR	GBP		8,35
	TRANSACTION	REFDATE	BASE_AMOUNT	BASE_CCY	EUR	USD
1	1	17/04/2014	10 EUR		10	13,56
2	2	05/04/2014	14 EUR		14	18,98
3	3	24/03/2014	14 EUR		16	20,78
4	4	18/03/2014	10 EUR		10	13,56
						8,35

### 5.31.1.5. Comment

- The column used as key for grouping information is the **TRANSACTION** column.
- The initial column whose values are transformed into columns in the output table is the **CALC\_CCY** column.
- The values of the initial table which are written into the newly created columns are those of the **CALC\_AMOUNT** column.

## 5.31.2. PROC TRANSPOSE - SingleCurrency Transactions

### 5.31.2.1. Goal

Get the values of the initial table, transpose them into columns in the output table, and write selected values into these tables.

### 5.31.2.2. Features being illustrated

- PROC TRANSPOSE2 step

### 5.31.2.3. Program

```
LIBNAME USER ":";
```

```
//Build the table to be transposed
```

```
DATA WORK.TOTRANSPOSE;
```

```
COLUMN TRANSACTION TYPE=INTEGER;
```

```
COLUMN REFDATE TYPE=DATE;
```

```
COLUMN BASE_AMOUNT TYPE=FLOAT;
```

```
COLUMN BASE_CCY TYPE=STRING;
```

```
COLUMN CALC_CCY TYPE=STRING;
```

```
COLUMN CALC_AMOUNT TYPE=FLOAT;
```

```
TRANSACTION=1;REFDATE="17/04/2014";BASE_AMOUNT=10;BASE_CCY="EUR";CALC_CCY="EUR";CALC_AMOUNT=10;OUTPUT;
```

```
TRANSACTION=2;REFDATE="05/04/2014";BASE_AMOUNT=14;BASE_CCY="EUR";CALC_CCY="EUR";CALC_AMOUNT=14;OUTPUT;
```

```
TRANSACTION=3;REFDATE="24/03/2014";BASE_AMOUNT=14;BASE_CCY="EUR";CALC_CCY="EUR";CALC_AMOUNT=16;OUTPUT;
```

```
TRANSACTION=4;REFDATE="18/03/2014";BASE_AMOUNT=10;BASE_CCY="EUR";CALC_CCY="EUR";CALC_AMOUNT=10;OUTPUT;
```

```
RUN;
```

```
//Print the table to be transposed
```

```
PROC PRINT DATA = WORK.TOTRANSPOSE;
```

RUN;

//Transpose the columns of the data table

```
PROC TRANSPOSE2 DATA=WORK.TOTRANSPOSE  
OUT=WORK.TABLE_TRANSPOSED NAME=COLUMNS LABEL="Columns";
```

```
VAR REFDATE BASE_AMOUNT BASE_CCY CALC_CCY CALC_AMOUNT;
```

```
ID TRANSACTION;
```

```
IDLABEL TRANSACTION;
```

RUN;

//Print the resulting table

```
PROC PRINT DATA=WORK.TABLE_TRANSPOSED LABEL;
```

RUN;

#### 5.31.2.4. Result

◀ [PROC-TRANSPOSE\\_SingleCurrencyTransactions](#)

Execute						
	TRANSACTION	REFDATE	BASE_AMOUNT	BASE_CCY	CALC_CCY	CALC_AMOUNT
1	1	17/04/2014		10 EUR	EUR	10
2	2	05/04/2014		14 EUR	EUR	14
3	3	24/03/2014		14 EUR	EUR	16
4	4	18/03/2014		10 EUR	EUR	10
Columns		1	2	3	4	
1	REFDATE	17/04/2014	05/04/2014	24/03/2014	18/03/2014	
2	BASE_AMOUNT	10	14	14	10	
3	BASE_CCY	EUR	EUR	EUR	EUR	
4	CALC_CCY	EUR	EUR	EUR	EUR	
5	CALC_AMOUNT	10	14	16	10	

### 5.31.2.5. Comment

- The column used as key for grouping information is the **TRANSACTION** column. The values of this column are used to create the columns of the output table.
- A new column entitled **Columns** has been defined in the output table, which contains the labels of the initial columns.
- All the columns of the initial table are transposed into lines.

## 5.32. Examples - PROC UPDATE

The following examples illustrate the **PROC UPDATE** step and the difference between **PROC UPDATE** and **DATA MERGE**:

- [PROC UPDATE - Merge two cubes](#)
- [PROC UPDATE and DATA MERGE - Comparison](#)

### 5.32.1. PROC UPDATE - Merge two cubes

#### 5.32.1.1. Goal

Create an eFront Cube cube from two different source cubes.

#### 5.32.1.2. Features being illustrated

- [PROC UPDATE step](#)
- [PROC FAQUERY step](#)

#### 5.32.1.3. Program

LIBNAME User ".";

//Create a global cube from a QueryBuilder query

PROC FAQUERY QUERY="FIA405 - QueryBuilderSamples\CRM - Fund Investors";

TABLE "SummaryPerInvestors" OUT=SummaryPerInvestors;

RUN;

//For illustration purpose, print the cube before merge

PROC PRINT DATA=SummaryPerInvestors;

RUN;

//Merge the global cube with another global cube which exists already on the FrontCube Server

PROC UPDATE DATA=SummaryPerInvestors WITH=ALLFUNDS;

CLASS IQID;

RUN;

//For illustration purpose, print the merged cube

PROC PRINT DATA=SummaryPerInvestors;

RUN;

### 5.32.1.4. Result before the merge

The first cube built from the QueryBuilder (before the merge):

[BuildMergedCube](#)

Execute								
	IQID	FUND	CURRENCY	INVESTOR	INVESTOR_CURREN	COMMITMENT	TOTAL_CALLED	TOTAL_DISTRIBUTE
1	AB8F9B86F48448D1A		EUR	Investor 1	EUR	20000000,0000	3000000,0000	
2	AB8F9B86F48448D1A		EUR	Investor 2	EUR	20000000,0000	3000000,0000	
3	AB8F9B86F48448D1A		EUR	Investor 3		700000,0000		
4	AB8F9B86F48448D1A		EUR	Investor 4	EUR	10000000,0000		
5	8683B27383AC446FB		EUR	Investor 5		30000000,0000	4500000,0000	
6	8683B27383AC446FB		EUR	Investor 6	EUR	2000000,0000	300000,0000	
7	8683B27383AC446FB		EUR	Investor 7	EUR	7000000,0000		
8	8683B27383AC446FB		EUR	Investor 8	EUR	800000,0000	120000,0000	
9	8683B27383AC446FB		EUR	Investor 9	EUR			
10	2177E714887247F2Fund F		EUR	LP 1	EUR			
11	2177E714887247F2Fund F		EUR	LP 2				
12	2177E714887247F2Fund F		EUR	LP 3				
13	035F802539D640C6 Import_Fund		EUR	Import_investor1		25000000,0000	2000000,0000	
14	035F802539D640C6 Import_Fund		EUR	Import_investor2			1000000,0000	
15	EF227B6A02D5490 Import_Fund_PE_wt EUR			Import_Investor1		25000000,0000	1557415,0000	3457183,0000
16	FD2C363AB481429 Import_Fund_PE_wo EUR			Import_Investor2		23000000,0000	375000,0000	81385,4600
17	082C8F0719734D2E MF1 - Managed PE	EUR		Investor 1	EUR	0,0000	-115558,5000	205202,7700
18	082C8F0719734D2E MF1 - Managed PE	EUR		Investor 2	USD	3925839,188450781248608,646894528	638413,3092395077	
19	082C8F0719734D2E MF1 - Managed PE	EUR		Investor 3	USD	3906586,56727981779088,0700	2234288,823859261	
20	082C8F0719734D2E MF1 - Managed PE	EUR		Late coming				
21	05C5B798F05B4CB1MF2 - Managed PE	EUR		Late-coming				
22	05C5B798F05B4CB1MF2 - Managed PE	EUR		Investor 1	EUR	0,0000	32157,3800	47890,4200
23	05C5B798F05B4CB1MF2 - Managed PE	EUR		Investor 2	USD	2616414,265320865119381,0000	110696,2700	
24	05C5B798F05B4CB1MF2 - Managed PE	EUR		Investor 3	USD	3924758,902291509180020,9100	166044,3800	
25	05C5B798F05B4CB1MF2 - Managed PE	EUR		Investor 4	EUR	1000000,0000	23078,5700	35571,4400
26	05C5B798F05B4CB1MF2 - Managed PE	EUR		Investor 5	EUR	1000000,0000	7142,8600	

### 5.32.1.5. Result after the merge

The initial cube after the merge. The IQID column is used to merge the cubes.

#### ›BuildMergedCube

Execute										
	IQID	FUND	CURRENCY	INVESTOR	INVESTOR_CURREN	COMMITMENT	TOTAL_CALLED	TOTAL_DISTRIBUTE	ABBREVIATION	
1	ABBF9B86F48448DFA		EUR	Investor 1	EUR	20000000,0000	3000000,0000		A	
2	ABBF9B86F48448DFA		EUR	Investor 2	EUR	20000000,0000	3000000,0000		A	
3	ABBF9B86F48448DFA		EUR	Investor 3		700000,0000			A	
4	ABBF9B86F48448DFA		EUR	Investor 4	EUR	10000000,0000			A	
5	8683B27383AC446FB		EUR	Investor 5		30000000,0000	4500000,0000		B	
6	8683B27383AC446FB		EUR	Investor 6	EUR	2000000,0000	300000,0000		B	
7	8683B27383AC446FB		EUR	Investor 7	EUR	7000000,0000			B	
8	8683B27383AC446FB		EUR	Investor 8	EUR	800000,0000	120000,0000		B	
9	8683B27383AC446FB		EUR	Investor 9	EUR				B	
10	2177E714887247F2!Fund F		EUR	LP 1	EUR					
11	2177E714887247F2!Fund F		EUR	LP 2						
12	2177E714887247F2!Fund F		EUR	LP 3						
13	035F802538D640C6 Import_Fund		EUR	Import_investor1		25000000,0000	2000000,0000		Imp_Fund	
14	035F802538D640C6 Import_Fund		EUR	Import_investor2			1000000,0000		Imp_Fund	
15	EF227B6A02D5490C Import_Fund_PE_w/o EUR			Import_investor1		25000000,0000	1557415,0000	3457183,0000		
16	FD2C363AB481429E Import_Fund_PE_w/o EUR			Import_investor2		23000000,0000	375000,0000	81385,4600		
17	082C8F0719734D2E MF1 - Managed PE	EUR		Investor 1	EUR	0,0000	-115558,5000	205202,7700	Managed PE Fund	
18	082C8F0719734D2E MF1 - Managed PE	EUR		Investor 2	USD	3925839,188450781248608,646894528	638413,3092395077	Managed PE Fund		
19	082C8F0719734D2E MF1 - Managed PE	EUR		Investor 3	USD	3906586,56727981779088,0700	2234288,823859261	Managed PE Fund		
20	082C8F0719734D2E MF1 - Managed PE	EUR		Late coming					Managed PE Fund	
21	05C5B798F05B4CB!MF2 - Managed PE	EUR		Late-coming					MF2	
22	05C5B798F05B4CB!MF2 - Managed PE	EUR		Investor 1	EUR	0,0000	32157,3800	47890,4200	MF2	
23	05C5B798F05B4CB!MF2 - Managed PE	EUR		Investor 2	USD	2616414,265320865119381,0000	110696,2700	MF2		
24	05C5B798F05B4CB!MF2 - Managed PE	EUR		Investor 3	USD	3924758,902291509180020,9100	166044,3800	MF2		
25	05C5B798F05B4CB!MF2 - Managed PE	EUR		Investor 4	EUR	1000000,0000	23078,5700	35571,4400	MF2	

### 5.32.1.6. Comment

In comparison with the DATA MERGE statement, the PROC UPDATE step has a much better performance when creating merged cubes, as the impacted columns are not recalculated.

## 5.32.2. PROC UPDATE and DATA MERGE - Comparison

Although both the **PROC UPDATE** step and the **DATA MERGE** statement can be used to merge two cubes, they produce different outputs. For example, if 1 row in the first cube matches N rows in the right cube:

- when using **DATA MERGE**, the system outputs N rows in the merged cube
- when using **PROC UPDATE**, the system outputs 1 row in the merged cube

### 5.32.2.1. Features being illustrated

- **PROC UPDATE**
- **DATA MERGE**

### 5.32.2.2. Program

// Create a session cube with two investors:

DATA SESSION.INVESTOR;

COLUMN InvestorId;

COLUMN InvestorName;

InvestorId = "Inv1";

InvestorName = "Investor A";

OUTPUT;

InvestorId = "Inv2";

InvestorName = "Investor B";

OUTPUT;

RUN;

// Print the cube in order to compare it the with the results of the merge

PROC PRINT DATA=SESSION.INVESTOR; TITLE "Investors";

RUN;

// Create another session cube with four investees, two per each investor:

DATA SESSION.INVESTED;

COLUMN InvestedId;

COLUMN InvestorId;

COLUMN InvestedName;

InvestedId = "Comp1";

InvestorId = "Inv1";

InvestedName = "Company A";

OUTPUT;

InvestedId = "Comp2";

InvestorId = "Inv1";

InvestedName = "Company B";

OUTPUT;

InvestedId = "Comp3";

InvestorId = "Inv2";

InvestedName = "Company C";

OUTPUT;

InvestedId = "Comp4";

InvestorId = "Inv2";

InvestedName = "Company D";

```
OUTPUT;  
  
RUN;  
  
// Print the cube in order to compare it the with the results of the merge  
  
PROC PRINT DATA=SESSION.INVESTED; TITLE "Investees";  
  
RUN;  
  
// Merge the two source cubes by using DATA MERGE  
  
DATA SESSION.MERGED;  
  
MERGE SESSION.INVESTOR (IN=I1) SESSION.INVESTED;  
  
By InvestordId;  
  
IF I1 THEN  
  
OUTPUT;  
  
END;  
  
RUN;  
  
// Print the merged cube  
  
PROC PRINT DATA=SESSION.MERGED; TITLE "DATA MERGE";  
  
RUN;  
  
// Merge the two source cubes by using PROC UPDATE  
  
PROC UPDATE DATA=SESSION.INVESTOR WITH=SESSION.INVESTED;  
  
CLASS InvestordId;  
  
RUN;  
  
// Print the merged cube
```

```
PROC PRINT DATA=SESSION.INVESTOR; TITLE "PROC UPDATE";  
RUN;
```

### 5.32.2.3. Results before the merge

### 5.32.2.4.

### 5.32.2.5. Results after the merge

The Investorld column is used to merge the cubes.

### 5.32.2.6.

## 5.33. Examples - STYLESHEET step

The following examples illustrate the PROC STYLESHEET step, and subordinate statements:

- STYLESHEET - PROC TABULATE (1)
- STYLESHEET - PROC TABULATE (2)
- STYLESHEET - PROC PRINT

### 5.33.1. STYLESHEET - PROC TABULATE (1)

#### 5.33.1.1. Goal

Apply tow sets of styles to format an output table : customized styles + default styles.

#### 5.33.1.2. Features being illustrated

- STYLESHEET step
- PROC TABULATE step
- Text style attributes and values

#### 5.33.1.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

// Define styles for the PROC TABULATE step

```
STYLESHEET;
```

```
TABULATE(TITLE) = (ALIGN=CENTER PADDING="8px");
```

```
TABULATE(TITLE1) = (ALIGN=RIGHT PADDING="8px");
```

```
TABULATE(DATA) = (BACKGROUND=LIGHTBLUE FOREGROUND=WHITE  
FONT_FACE="Tahoma"
```

```
FONT_SIZE="9pt" FONT_WEIGHT="Bold" FONT_STYLE="Italic" ALIGN=RIGHT);
```

```
TABULATE(CLASS) = (BACKGROUND=GREEN FOREGROUND=WHITE  
FONT_FACE="Tahoma"  
  
FONT_SIZE="9pt" FONT_WEIGHT="Bold" FONT_STYLE="Italic" ALIGN=RIGHT);  
  
TABULATE(HEADER) = (BACKGROUND=BLUE FOREGROUND=WHITE  
FONT_FACE="Tahoma"  
  
FONT_SIZE="9pt" FONT_WEIGHT="Bold" FONT_STYLE="Italic" ALIGN=RIGHT);  
  
TABULATE(BOX) = (BACKGROUND="" BORDER="none");  
  
STYLESHEET END;  
  
//Define the statistical data table using the PROC TABULATE step  
  
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;  
  
CLASS USER_ID;  
  
VAR UNIT_PRICE;  
  
TABLE USER_ID = "Customer" ALL="Total"),  
  
UNIT_PRICE = "Items" *  
  
(N = "Number of Items" SUM = "Total Amount");  
  
RUN;  
  
//Re-initialize the stylesheet  
  
STYLESHEET;  
  
STYLESHEET END;  
  
//Display the above data table again  
  
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;  
  
CLASS USER_ID;
```

```

VAR UNIT_PRICE;

TABLE USER_ID = "Customer" ALL="Total"),
UNIT_PRICE = "Items"*
(N = "Number of Items" SUM = "Total Amount");
RUN;

```

#### 5.33.1.4. Result

		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
	Total	8	53423,1

		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
	Total	8	53423,1

#### 5.33.2. STYLESHEET - PROC TABULATE (2)

##### 5.33.2.1. Goal

Apply a stylesheet to a PROC TABULATE step.

##### 5.33.2.2. Features being illustrated

- STYLESHEET statement
- PROC TABULATE step

- [Text style attributes and values](#)

### 5.33.2.3. Program

```
//Create the alias that points the existing table location  
  
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
  
/// Define styles for the PROC TABULATE step  
  
STYLESHEET STYLE_1;  
  
TABULATE(TITLE) = (ALIGN=CENTER PADDING="8px");  
  
TABULATE(TITLE1) = (ALIGN=RIGHT PADDING="8px");  
  
TABULATE(DATA) = (BACKGROUND=GRAY FOREGROUND=WHITE  
FONT_FACE="Tahoma"  
  
FONT_SIZE="9pt" FONT_WEIGHT="Bold" FONT_STYLE="Italic" ALIGN=RIGHT);  
  
TABULATE(CLASS) = (BACKGROUND=GREEN FOREGROUND=WHITE  
FONT_FACE="Tahoma"  
  
FONT_SIZE="9pt" FONT_WEIGHT="Bold" FONT_STYLE="Italic" ALIGN=RIGHT);  
  
TABULATE(HEADER) = (BACKGROUND=BLUE FOREGROUND=WHITE  
FONT_FACE="Tahoma"  
  
FONT_SIZE="9pt" FONT_WEIGHT="Bold" FONT_STYLE="Italic" ALIGN=RIGHT);  
  
TABULATE(BOX) = (BACKGROUND="" BORDER="none");  
  
STYLESHEET END;  
  
//Apply the default styles to the first PROC TABULATE step  
  
STYLESHEET;  
  
STYLESHEET END;  
  
//Define the statistical data table using the PROC TABULATE step
```

```
PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;
CLASS USER_ID;
VAR UNIT_PRICE;
TABLE USER_ID = "Customer" ALL="Total",
UNIT_PRICE = "Items"*
(N = "Number of Items" SUM = "Total Amount");
RUN;

//Apply the customized stylesheet to the second PROC TABULATE step

PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS
STYLE=STYLE_1;
CLASS USER_ID;
VAR UNIT_PRICE;
TABLE USER_ID = "Customer" ALL="Total",
UNIT_PRICE = "Items"*
(N = "Number of Items" SUM = "Total Amount");
RUN;
```

### 5.33.2.4. Result

		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
<b>Total</b>		8	53423,1

		Items	
		Number of Items	Total Amount
Customer	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
<b>Total</b>		8	53423,1

### 5.33.2.5. Comment

- To call an existing stylesheet, the stylesheet definition must precede the stylesheet calling PROC step.

## 5.33.3. STYLESHEET - PROC PRINT

### 5.33.3.1. Goal

Apply a stylesheet to a PROC PRINT step.

### 5.33.3.2. Features being illustrated

- STYLESHEET statement
- PROC PRINT step
- Text style attributes and values

### 5.33.3.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
STYLESHEET [eFront_styles]  
STYLE=(ALIGN=LEFT VALIGN=TOP BACKGROUND="cornsilk");  
PRINT(TITLE)=(BACKGROUND="aqua" FONT_WEIGHT="BOLD");  
PRINT(TITLE1)=(METALLIC);  
PRINT(GRANDTOTAL)=(FONT_WEIGHT="BOLD");  
PRINT(DATA)=(BACKGROUND="aqua");  
PRINT(HEADER)=(VALIGN=TOP);  
PRINT(N)=(VALIGN=VCENTER);  
PRINT(OBS)=(FONT_WEIGHT="BOLD" ALIGN=LEFT);  
PRINT(COLHEADER)=(ALIGN=RIGHT FOREGROUND="crimson");  
PRINT(TOTAL)=(FONT_WEIGHT="NORMAL");  
STYLESHEET END;  
//Create a personalized format  
PROC FORMAT;  
PICTURE P_Money  
LOW-HIGH = '$ 00.00';  
RUN;  
//Sort the table  
PROC SORT DATA=TEST.CONTRACTS OUT=TEST.CONTRACTS_SORTED  
NODUPKEY;
```

```
BY USER_ID;  
RUN;  
//Display data on screen  
PROC PRINT DATA=TEST.CONTRACTS_SORTED LABEL STYLE="eFront_styles";  
RUN;
```

#### 5.33.3.4. Result

#### 5.33.3.5.

#### 5.33.3.6. Comment

- To call an existing stylesheet, the stylesheet definition must precede the stylesheet calling PROC step.

## 5.34. Examples - global statements

The following examples illustrate the global statements:

- LIBNAME\_Magic\_Variables
- LIBNAME - USER - CURRENT

### 5.34.1. LIBNAME - Magic variables

#### 5.34.1.1. Goal

Set up the standard Efront librairies to be referred to during an **eFront Script** working session.

#### 5.34.1.2. Features being illustrated

- LIBNAME statement
- Magic variables

#### 5.34.1.3. Program

//Create standard directories

```
LIBNAME ST "\{SHARED}\FV\STANDARD";
```

```
LIBNAME ST_DM "\{SHARED}\FV\STANDARD\DATAMART";
```

```
LIBNAME ST_DM_TB "\{SHARED}\FV\STANDARD\DATAMART\TABLES";
```

```
LIBNAME ST_DM_PG "\{SHARED}\FV\STANDARD\DATAMART\PROGRAMS";
```

//Create aliases pointing to the standard eFrontFolders

```
LIBNAME TEMP "\{PRIVATE}\eFront\temp";
```

```
LIBNAME SHARED_TEMP "\{SHARED}\eFront\temp";
```

```
LIBNAME SITE "\{SHARED}\eFront\site\FV";
```

### 5.34.1.4. Comment

- The LIBNAME statements must precede the programming steps.
- Magic variables are being used.

## 5.34.2. LIBNAME - USER - CURRENT

### 5.34.2.1. Goal

Write a table to the current work folder

### 5.34.2.2. Features being illustrated

- LIBNAME statement

### 5.34.2.3. Program

### 5.34.2.4.

```
LIBNAME USER CURRENT;
DATA TableSimple;
COLUMN Id;
COLUMN FirstName;
```

### 5.34.2.5. Comment

- Creates the table produced by the program within the current folder of the user.

## 5.35. Examples - macro statements and controls

The following examples illustrate macro statements, and macro controls:

- %DEFINE
- %FOR - %NEXT - MOD
- %INCLUDE
- %INCLUDE - %PARAM - %IF %THEN %END
- %LET - %DO WHILE - %LOOP WHILE
- %LET - %FOR - %NEXT
- %LET - %FOR - %NEXT - @TABLE
- %LET - %SELECT - %WHEN - %END
- %LET - %WHILE - %END
- %LET - INSTR()
- %LET - LINKFILES() - LINKFILE()
- %PARAM
- %PARAM - ??CHOICE - LIKE
- %PARAM - ?? ID - WHERE
- %PARAM - LINKFILESEXT() - LINKFILE()
- %PARAM\_TYPE\_DEFAULT
- %PARAM - ??LOOKUP - ??XLOOKUPCODE
- %PARAM - ??PICKID(SQL- query) - ??FILTER
- %PARAM - ??PICKID\_{VAR}
- %PARAM - ??PICKID\_{VAR}
- %PARAM - ??XPICKID - SHOWSELECTEDITEMS
- %PARAM - ??XPICKID with SQL query

### 5.35.1. %DEFINE

#### 5.35.1.1. Goal

Execute an **eFront Script** program, if a particular macro exists.

#### 5.35.1.2. Features being illustrated

- %DEFINE statement

- [Functions - special](#)

### 5.35.1.3. Program

```
/*Test the existence of particular macros*/  
  
%IF EXISTMACRO("BUILD_PORTCOS") OR EXISTMACRO("PORTFOLIO_REPORT")  
%THEN  
  
%DEFINE BUILD_PORTCOS;  
  
%END;  
  
%IF EXISTMACRO("BUILD_FUNDS") OR  
EXISTMACRO("FOF_INVESTOR_PORTFOLIO_REPORT") %THEN  
  
%DEFINE BUILD_FUNDS;  
  
%END;  
  
%IF EXISTMACRO("BUILD_PORTCOS")%THEN  
  
%INCLUDE "FV_P_PORTCO_TRANSACTIONS";  
  
%END;  
  
%IF EXISTMACRO("BUILD_FUNDS")%THEN  
  
%INCLUDE "FV_P_FUND_TRANSACTIONS";  
  
%END;
```

### 5.35.1.4. Result

According to the macro being detected, different programs are run.

## 5.35.2. %FOR - %NEXT - MOD

### 5.35.2.1. Goal

Execute groups of statements according to a macro variables value.

### 5.35.2.2. Features being illustrated

- %FOR...%NEXT
- Operators
- TRACE statement

### 5.35.2.3. Program

```
TRACE "TEST FOR LOOP USING MOD";
```

```
%FOR %I =1 %TO 10
```

```
%IF (%I MOD 2) = 0 %THEN
```

```
%CONTINUE;
```

```
TRACE "FOR EVEN=" & %I;
```

```
%ELSE
```

```
TRACE "FOR ODD=" & %I;
```

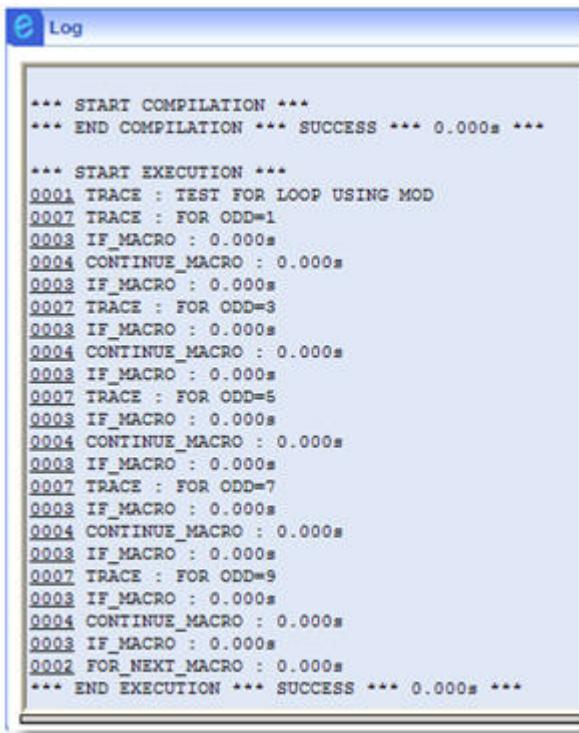
```
%END;
```

```
%NEXT;
```

### 5.35.2.4. Comment

- The operator & is used to concatenate the text string and the macro variable without any empty space in between.

### 5.35.2.5. Result



The screenshot shows a window titled "Log" with the following text content:

```
*** START COMPILED ***
*** END COMPILED *** SUCCESS *** 0.000s ***

*** START EXECUTION ***
0001 TRACE : TEST FOR LOOP USING MOD
0007 TRACE : FOR ODD=1
0003 IF_MACRO : 0.000s
0004 CONTINUE_MACRO : 0.000s
0003 IF_MACRO : 0.000s
0007 TRACE : FOR ODD=3
0003 IF_MACRO : 0.000s
0004 CONTINUE_MACRO : 0.000s
0003 IF_MACRO : 0.000s
0007 TRACE : FOR ODD=5
0003 IF_MACRO : 0.000s
0004 CONTINUE_MACRO : 0.000s
0003 IF_MACRO : 0.000s
0007 TRACE : FOR ODD=7
0003 IF_MACRO : 0.000s
0004 CONTINUE_MACRO : 0.000s
0003 IF_MACRO : 0.000s
0007 TRACE : FOR ODD=9
0003 IF_MACRO : 0.000s
0004 CONTINUE_MACRO : 0.000s
0003 IF_MACRO : 0.000s
0002 FOR_NEXT_MACRO : 0.000s
*** END EXECUTION *** SUCCESS *** 0.000s ***
```

### 5.35.3. %INCLUDE

#### 5.35.3.1. Goal

Include a program in a program.

#### 5.35.3.2. Features being illustrated

- [%INCLUDE statement](#)

#### 5.35.3.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
```

```
/*Include the program that generates and displays the original table*/
```

```
%INCLUDE "P_USERS";

//Define the statistical data table

PROC TABULATE DATA=TEST.USERS_INFO_ABOUT_CONTRACTS;

CLASS USER_ID;

VAR UNIT_PRICE;

TABLE USER_ID = "Customer" ALL = "Total",

UNIT_PRICE = "Items"*

(N = "Number of Items" SUM = "Total Amount");

RUN;
```

### 5.35.3.4. Result

	<b>Id</b>	<b>First Name</b>	<b>Name</b>	<b>Birthday</b>	<b>Children</b>	<b>Zip Code</b>	<b>City</b>
<b>1</b>	101	Bobby	Schmidt	05/03/1959	2	10021	New York
<b>2</b>	102	Ruber	Zodi	25/01/1980	1	75001	Paris
<b>3</b>	103	Deborah	Woodpecker	04/11/1954	5	33000	Bordeaux
<b>4</b>	104	Sandra	Specker	01/12/1962	4	33000	Bordeaux
<b>5</b>	105	Marjorie	Pattern	12/03/1983	0	10021	New York

	<b>Items</b>		<b>Total Amount</b>
	<b>Number of Items</b>		
<b>Customer</b>	101	4	19450,3
	102	2	6441,6
	103	1	7501
	105	1	20030,2
<b>Total</b>		<b>8</b>	<b>53423,1</b>

### 5.35.4. %INCLUDE - %PARAM - %IF %THEN %END

#### 5.35.4.1. Goal

Include a program in a program.

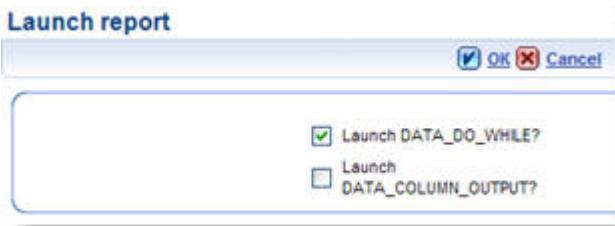
### 5.35.4.2. Features being illustrated

- %INCLUDE statement
- %PARAM statement
- %IF...%THEN...%END

### 5.35.4.3. Program

```
%PARAM EXECUTE_DATA_DO_WHILE TYPE=BOOLEAN LABEL="Launch  
DATA_DO_WHILE?" DEFAULT=FALSE;  
  
%PARAM EXECUTE_DATA_COLUMN_OUTPUT TYPE=BOOLEAN LABEL="Launch  
DATA_COLUMN_OUTPUT?" DEFAULT=FALSE;  
  
%IF%EXECUTE_DATA_DO_WHILE %THEN  
  
%INCLUDE "\\\EFRONT_SUP.2(Private)  
\TEST\EXAMPLES\Programs\DATA_DO_WHILE";  
  
%END;  
  
%IF%EXECUTE_DATA_COLUMN_OUTPUT %THEN  
  
%INCLUDE "\\\EFRONT_SUP.2(Private)  
\TEST\EXAMPLES\Programs\DATA_COLUMN_OUTPUT";  
  
%END;
```

### 5.35.4.4. Result



According to the choice being made, **eFront Script** launches the corresponding program.

R	
1	obert
2	bert
3	ert
4	rt
5	t

## 5.35.5. %LET - %DO WHILE - %LOOP WHILE

### 5.35.5.1. Goal

Execute groups of statements according to a macro variables value.

### 5.35.5.2. Features being illustrated

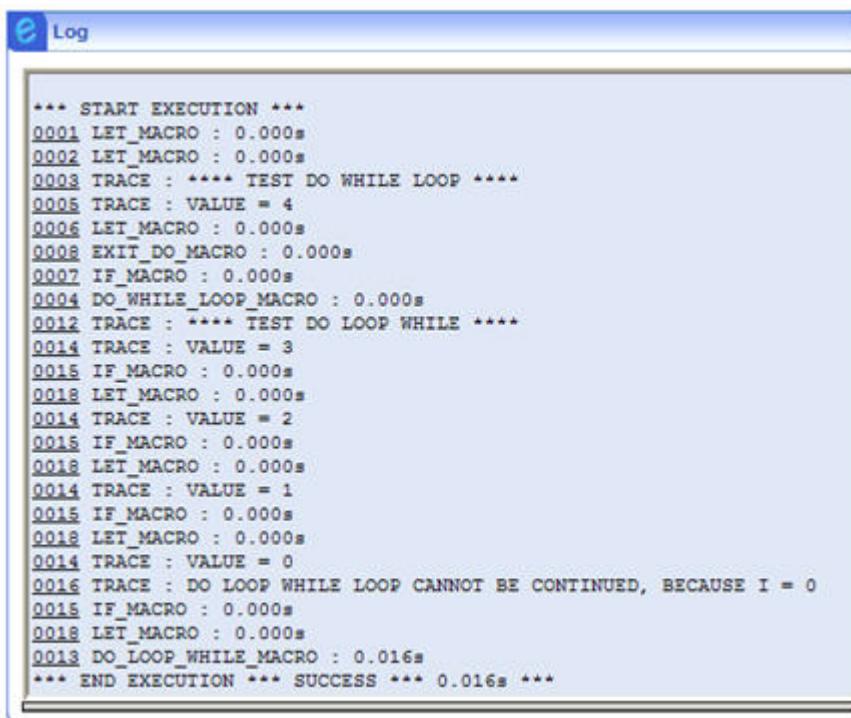
- %DO WHILE...%LOOP WHILE
- %LET statement
- TRACE statement

### 5.35.5.3. Program

```
%LET TEST = "VALUE = ";  
  
%LET I = 4;  
  
TRACE "**** TEST DO WHILE LOOP ****";  
  
%DO WHILE %I > 0  
  
TRACE %TEST %I;  
  
%LET I = %I-1;  
  
%IF %I = 3 %THEN  
  
%EXIT %DO;  
  
%END;
```

```
%LOOP;  
  
TRACE "**** TEST DO LOOP WHILE ****";  
  
%DO  
  
TRACE %TEST %I;  
  
%IF %I=0 %THEN  
  
TRACE "DO LOOP WHILE LOOP CANNOT BE CONTINUED, BECAUSE I = " %I;  
  
%END;  
  
%LET I = %I-1;  
  
%LOOP WHILE %I >= 0;
```

#### 5.35.5.4. Result



The screenshot shows a window titled 'Log' with a blue header bar. The main area contains a text log of script execution. The log starts with '\*\*\* START EXECUTION \*\*\*' and ends with '\*\*\* END EXECUTION \*\*\* SUCCESS 0.016s \*\*\*'. It includes numerous trace statements, macro assignments, and conditional checks, illustrating the flow of the script as it attempts to execute a do-while loop that cannot be continued due to the condition %I=0.

```
*** START EXECUTION ***
0001 LET_MACRO : 0.000s
0002 LET_MACRO : 0.000s
0003 TRACE : **** TEST DO WHILE LOOP ****
0005 TRACE : VALUE = 4
0006 LET_MACRO : 0.000s
0008 EXIT_DO_MACRO : 0.000s
0007 IF_MACRO : 0.000s
0004 DO_WHILE_LOOP_MACRO : 0.000s
0012 TRACE : **** TEST DO LOOP WHILE ****
0014 TRACE : VALUE = 3
0015 IF_MACRO : 0.000s
0018 LET_MACRO : 0.000s
0014 TRACE : VALUE = 2
0015 IF_MACRO : 0.000s
0016 LET_MACRO : 0.000s
0014 TRACE : VALUE = 1
0015 IF_MACRO : 0.000s
0018 LET_MACRO : 0.000s
0014 TRACE : VALUE = 0
0016 TRACE : DO LOOP WHILE LOOP CANNOT BE CONTINUED, BECAUSE I = 0
0015 IF_MACRO : 0.000s
0018 LET_MACRO : 0.000s
0013 DO_LOOP_WHILE_MACRO : 0.016s
*** END EXECUTION *** SUCCESS 0.016s ***
```

## 5.35.6. %LET - %FOR - %NEXT

### 5.35.6.1. Goal

Execute groups of statements according to a macro variables value.

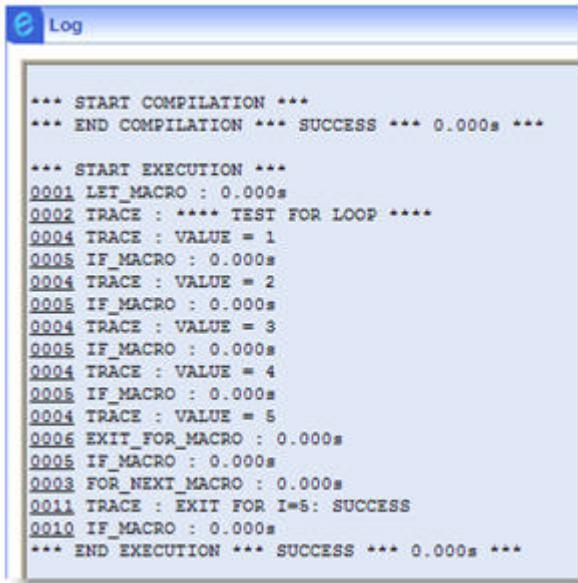
### 5.35.6.2. Features being illustrated

- %FOR...%NEXT
- %LET statement
- TRACE statement

### 5.35.6.3. Program

VERSION 1	VERSION 2
%LET TEST = "VALUE = ";	%LET TEST = "VALUE = ";
TRACE "**** TEST FOR LOOP ****";	TRACE "**** TEST FOR LOOP ****";
%FOR %I =1 %TO 10	%FOR %I %IN (1,2,3,4,5,6,7,8,9,10)
TRACE %TEST %I;	TRACE %TEST %I;
%IF %I = 5 %THEN	%IF %I = 5 %THEN
%EXIT %FOR;	%EXIT %FOR;
%END;	%END;
%NEXT;	%NEXT;
%IF %I = 5 %THEN	%IF %I = 5 %THEN
TRACE "EXIT FOR I=5:" "SUCCESS";	TRACE "EXIT FOR I=5:" "SUCCESS";
%ELSE	%ELSE
TRACE "EXIT FOR I=5:" "FAILURE";	TRACE "EXIT FOR I=5:" "FAILURE";
%END;	%END;

## 5.35.6.4. Result



```

Log

*** START COMPILED ***
*** END COMPILED *** SUCCESS *** 0.000s ***

*** START EXECUTION ***
0001 LET_MACRO : 0.000s
0002 TRACE : ***** TEST FOR LOOP *****
0004 TRACE : VALUE = 1
0005 IF_MACRO : 0.000s
0004 TRACE : VALUE = 2
0005 IF_MACRO : 0.000s
0004 TRACE : VALUE = 3
0005 IF_MACRO : 0.000s
0004 TRACE : VALUE = 4
0005 IF_MACRO : 0.000s
0004 TRACE : VALUE = 5
0006 EXIT_FOR_MACRO : 0.000s
0005 IF_MACRO : 0.000s
0003 FOR_NEXT_MACRO : 0.000s
0011 TRACE : EXIT FOR I=5: SUCCESS
0010 IF_MACRO : 0.000s
*** END EXECUTION *** SUCCESS *** 0.000s ***

```

## 5.35.7. %LET - %FOR - %NEXT - @TABLE

### 5.35.7.1. Goal

Creating tables repetitively, and referencing dynamically columns of the new tables.

### 5.35.7.2. Features being illustrated

- %FOR...%NEXT
- %LET statement
- Macro variables

### 5.35.7.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TableFormation";
```

```
%FOR %I=1 %TO 5
```

```
%LET TABLE = "TABLE_" & %I;
```

```
%LET VAR = "COL" & %I;  
  
DATA TEST.@TABLE;  
  
COLUMN COL1;  
  
COLUMN COL2;  
  
COLUMN COL3;  
  
COLUMN COL4;  
  
COLUMN COL5;  
  
@VAR = %I;  
  
OUTPUT;  
  
RUN;  
  
PROC PRINT DATA = TEST.@TABLE NOOBS STYLE(TITLE1)=(ALIGN="LEFT"  
FONT_SIZE="22");  
  
TITLE1 %TABLE;  
  
RUN;  
  
%NEXT;
```

#### 5.35.7.4. Comment

- As you can see, **@TABLE** references a dynamic table name, and **@VAR** references an existing column name.
- For the moment it is not possible, to create dynamically column names.
- To reference the column dynamically, you don't necessarily have to use the **%LET** statement, you can directly build an expression after the prefix **@**:

...

```
COLUMN COL5;  
@("COL" & %I) = %I;
```

OUTPUT;

...

### 5.35.7.5. Result

HTML Output Execute					
TABLE_1					
	COL1	COL2	COL3	COL4	COL5
1	1				
TABLE_2					
	COL1	COL2	COL3	COL4	COL5
1		2			
TABLE_3					
	COL1	COL2	COL3	COL4	COL5
1			3		
TABLE_4					
	COL1	COL2	COL3	COL4	COL5
1				4	
TABLE_5					
	COL1	COL2	COL3	COL4	COL5
1					5

## 5.35.8. %LET - %SELECT - %WHEN - %END

### 5.35.8.1. Goal

Execute groups of statements according to a macro variables value.

### 5.35.8.2. Features being illustrated

- %SELECT... %WHEN... %THEN...%END
- %LET statement
- TRACE statement

### 5.35.8.3. Program

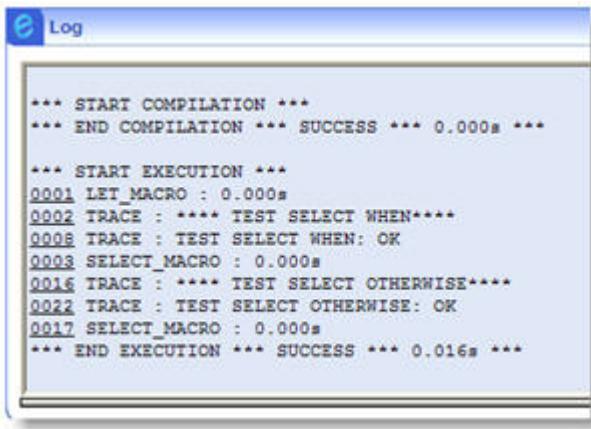
```
%LET I = 2;
```

```
TRACE "**** TEST SELECT WHEN****";
```

```
%SELECT %I
```

```
%WHEN 1 %THEN  
TRACE "FAILURE";  
%END;  
  
%WHEN 2 %THEN  
TRACE "TEST SELECT WHEN: OK";  
%END;  
  
%WHEN 3 %THEN  
TRACE "FAILURE";  
%END;  
%END;  
TRACE "**** TEST SELECT OTHERWISE****";  
%SELECT %I  
  
%WHEN 1 %THEN  
TRACE "FAILURE";  
%END;  
  
%OTHERWISE  
TRACE "TEST SELECT OTHERWISE: OK";  
%END;  
%END;
```

### 5.35.8.4. Result



The screenshot shows a window titled "Log" with the following text content:

```
*** START COMPILED ***
*** END COMPILED *** SUCCESS *** 0.000s ***

*** START EXECUTION ***
0001 LET_MACRO : 0.000s
0002 TRACE : **** TEST SELECT WHEN ****
0008 TRACE : TEST SELECT WHEN: OK
0003 SELECT_MACRO : 0.000s
0016 TRACE : **** TEST SELECT OTHERWISE ****
0022 TRACE : TEST SELECT OTHERWISE: OK
0017 SELECT_MACRO : 0.000s
*** END EXECUTION *** SUCCESS *** 0.016s ***
```

## 5.35.9. %LET - %WHILE - %END

### 5.35.9.1. Goal

Execute groups of statements according to a macro variables value.

### 5.35.9.2. Features being illustrated

- %WHILE...%END
- %LET statement
- TRACE statement

### 5.35.9.3. Program

```
%LET TEST = "VALUE = ";
```

```
%LET I = 5;
```

```
TRACE "**** TEST WHILE****";
```

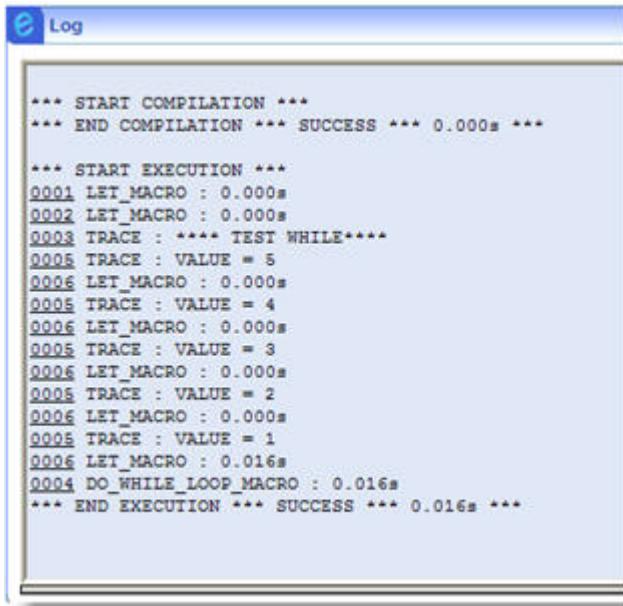
```
%WHILE %I > 0
```

```
TRACE %TEST %I;
```

```
%LET I = %I-1;
```

```
%END;
```

### 5.35.9.4. Result



The screenshot shows a window titled "Log" with the following text content:

```
*** START COMPILED ***
*** END COMPILED *** SUCCESS *** 0.000s ***

*** START EXECUTION ***
0001 LET_MACRO : 0.000s
0002 LET_MACRO : 0.000s
0003 TRACE : **** TEST WHILE ****
0004 TRACE : VALUE = 5
0005 LET_MACRO : 0.000s
0006 TRACE : VALUE = 4
0007 LET_MACRO : 0.000s
0008 TRACE : VALUE = 3
0009 LET_MACRO : 0.000s
0010 TRACE : VALUE = 2
0011 LET_MACRO : 0.000s
0012 TRACE : VALUE = 1
0013 LET_MACRO : 0.016s
0014 DO WHILE_LOOP_MACRO : 0.016s
*** END EXECUTION *** SUCCESS *** 0.016s ***
```

## 5.35.10. %LET - INSTR()

### 5.35.10.1. Goal

Search a table for particular values, and process according to the search result.

### 5.35.10.2. Features being illustrated

- [%LET statement](#)
- [INSTR\(\)](#)

### 5.35.10.3. Program

```
%LET C_INVESTMENTATCOST_PLUS=";SOU-ACT;SOU-ACT-CONV;SOU-BSA;SOU-BSA-CONV;SOU-OBL;SOU-OBLI-CONV;ACH-ACT;ACH-ACT-CONV;ACH-BSA;ACH-BSA-CONV;ACH-COU;ACH-COUPIK;ACH-OBL;ACH-OBLI-CONV;ECH-OBLIG;ECH-VENTE-OBLIG;VER-CCA;VER-PRET;VER-PRET-CNV;VER-PRET-CONV;CON-ACT;"
```

```
DATA WORK.BUY_SECURITY;
```

```
SET WORK.FV_COMPANY_TRANSACTION_FILTERED;
```

```
IF INSTR(%C_INVESTMENTATCOST_PLUS,";" ||  
TRIM(UCASE(TRANSACTION_TYPE)) || ":" ) <> 0 THEN  
  
_OUTPUT_= TRUE;  
  
ELSE  
  
_OUTPUT_= FALSE;  
  
END;  
  
RUN;
```

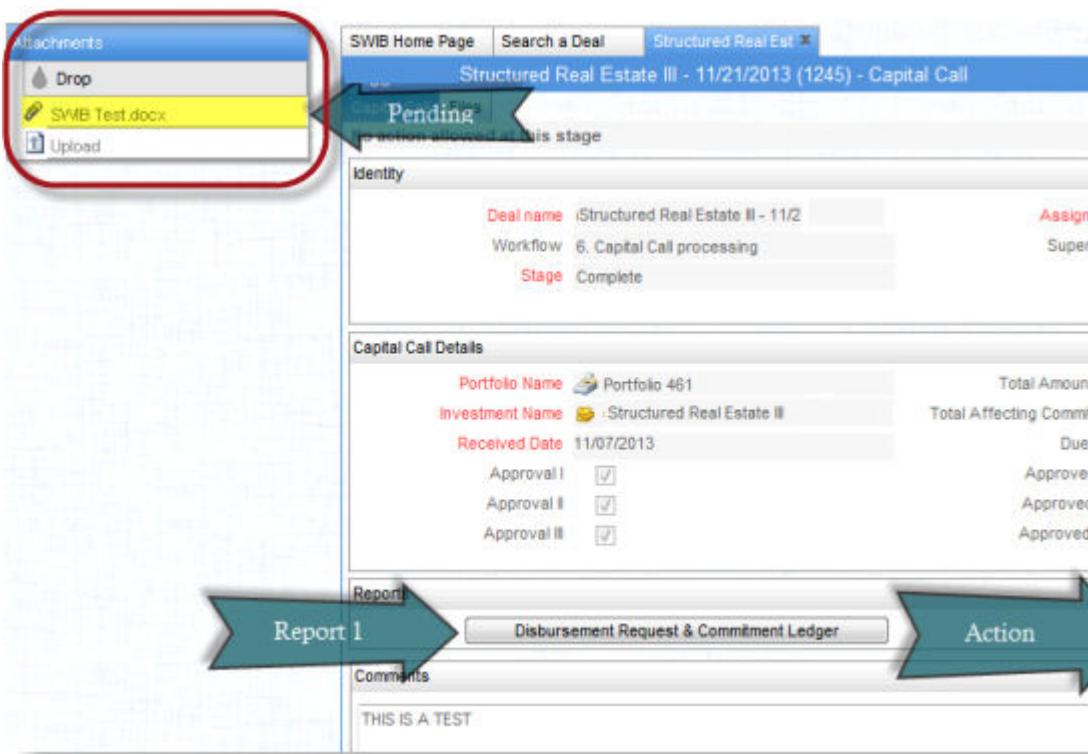
#### 5.35.10.4. Comment

- %LET is used to declare a group of strings to search for in a data table.
- As %LET only accepts one single character string, the first program step consists in extracting individual values from the string before proceeding to the evaluation of the condition.
- If you declare the group of values to search for using a %LET constant, and pointing to it from within the INSTR(), **eFront Script** searches the values directly in the data table, optimizing processing time.

#### 5.35.11. %LET - LINKFILES() - LINKFILE()

##### 5.35.11.1. Goal

Retrieve the file(s) in the **Attachments** section of a fund operation.



### 5.35.11.2. Features being illustrated

- %LET statement
- Functions - files and folders

### 5.35.11.3. Program

//Build the table that receives the uploaded image file

```
%Let ID = "0744A17F0D3D44AE81ACBEE1DEB201C4"; // Id of fund operation
```

```
%FOR%I = 1 %TO(LINKFILES(%ID))
```

```
Trace FILENAME(%I); // Display in the log the filename. Useful to check if you found the right file.
```

```
PROC PRINT;
```

```
PUT LINKFILE(%I);
```

RUN; // Give a "temporary" path where this file is available (this function has downloaded the file from admfile to the download directory)

```
%NEXT;
```

#### 5.35.11.4. Comment

- You may want to use this program to pull out the fund logos that have been uploaded for a fund.

### 5.35.12. %PARAM

#### 5.35.12.1. Goal

Use interactive user input to select table data.

#### 5.35.12.2. Features being illustrated

- [%PARAM statement](#)

#### 5.35.12.3. Program

```
//Create an alias
```

```
LIBNAME TEST "\EFront_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

```
%PARAM FIRSTNAME LABEL="Enter a first name";
```

```
//Create a table
```

```
DATA Names_UserInput;
```

```
COLUMN R TYPE=CHARLABEL="Name";
```

```
COLUMN I TYPE=INTEGER;
```

```
R=%FIRSTNAME;
```

```
I=0;
```

```
DO WHILE(I<5)
R=MID(R,2,5);
I=I+1;
OUTPUT;
LOOP;
RUN;
//Print the table
PROC PRINT DATA=Names_UserInput (DROP=I);
RUN;
```

#### 5.35.12.4. Comment

- %PARAM is used to declare a macro variable that receives its value through interactive user input.

#### 5.35.13. %PARAM - ??CHOICE - LIKE

##### 5.35.13.1. Goal

Use interactive user input to select data from a table.

##### 5.35.13.2. Features being illustrated

- %PARAM statement
- Operators

##### 5.35.13.3. Program

//Create an alias

```
LIBNAME TEST "\EFRONT_SUP.2/Private\TEST\EXAMPLES\TablesFormation";
//Create macro variable and assign values
```

```
%PARAM PREFIX LABEL = "Please choose a prefix" INFORMAT="???
CHOICE(AX,OF,BF,RV,PVI)";

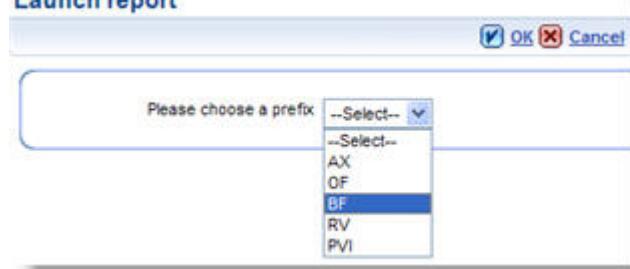
//Print Selected funds

PROC PRINT DATA = TEST.ALLFUNDS (WHERE (FUND LIKE (%PREFIX & "%")))
LABEL NOOBS;

RUN;
```

#### 5.35.13.4. Comment

[Launch report](#)



Fund	Closing date	Vintage year
BF_fund_1	05/01/2006	2003

#### 5.35.14. %PARAM - ??ID - WHERE

##### 5.35.14.1. Goal

Use interactive user input to select table data.

##### 5.35.14.2. Features being illustrated

- [%PARAM statement](#)
- [WHERE option](#)

##### 5.35.14.3. Program

```
%LET C_INVESTMENTATCOST_PLUS="";SOU-ACT;SOU-ACT-CONV;SOU-BSA;SOU-
BSA-CONV;SOU-OBL;SOU-OBLI-CONV;ACH-ACT;ACH-ACT-CONV;ACH-BSA;ACH-
```

```
BSA-CONV;ACH-COU;ACH-COUPIK;ACH-OBL;ACH-OBLI-CONV;ECH-OBLIG;ECH-  
VENTE-OBLIG;VER-CCA;VER-PRET;VER-PRET-CNV;VER-PRET-CONV;CON-ACT";  
  
%PARAM EMZFUND LABEL="FUNDS" INFORMAT="??ID(VCFUND)";  
  
DATA WORK.BUY_SECURITY;  
  
SET WORK.FV_COMPANY_TRANSACTION_FILTERED (WHERE  
INVESTOR_FUND_ID IN (%EMZFUND));  
  
IF INSTR(%C_INVESTMENTATCOST_PLUS,";" ||  
TRIM(UCASE(TRANSACTION_TYPE)) || ":" ) <> 0 THEN  
  
_OUTPUT_= TRUE;  
  
ELSE  
  
_OUTPUT_= FALSE;  
  
END;  
  
RUN;
```

#### 5.35.14.4. Comment

- %PARAM is used to declare a macro variable that receives its value through interactive user input.
- As %LET only accepts one single character string, the first program step consists in extracting individual values from the string before proceeding to the evaluation of the condition.

#### 5.35.15. %PARAM - LINKFILESEXT() - LINKFILE()

##### 5.35.15.1. Goal

Retrieve images that are attached to table data.

### 5.35.15.2. Features being illustrated

- `%PARAM` statement
- `LINKFILESEXT()`, `LINKFILE()`

### 5.35.15.3. Program

```
%PARAM PARAM_IDFUND LABEL="FUNDS" INFORMAT="??ID(VCFUND)";

DATA WORK.IMAGES(KEEP=FUND_ID FICHIER);

SET [SP_DW_TB].[EFFV_CDW_T_FUND](WHERE FUND_ID IN (%PARAM_IDFUND));

COLUMN FICHIER;

COLUMN I TYPE=INTEGER;

FOR I=1 TO LINKFILESEXT(FUND_ID,".PNG;.GIF")

FICHIER = LINKFILE(I);

OUTPUT;

NEXT;

RUN;
```

### 5.35.15.4. Comment

- `%PARAM` is used to declare a macro variable that receives its value through interactive user input.
- `LINKFILESEXT()` searches for the image files that are attached to the fund identified with `FUND_ID`. Each image file being attached to the fund is written to the `WORK.IMAGES` table.

## 5.35.16. %PARAM - TYPE - DEFAULT

### 5.35.16.1. Goal

Use interactive user input to select table data.

## 5.35.16.2. Features being illustrated

- %PARAM statement
- Functions - date

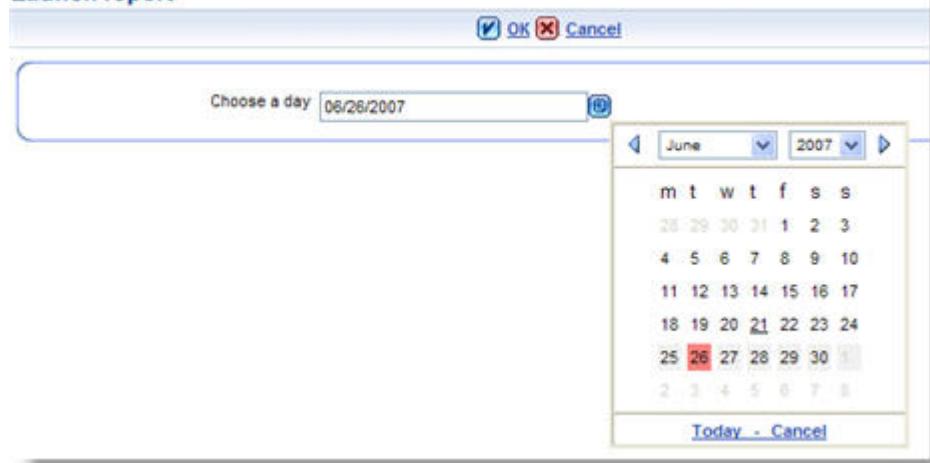
## 5.35.16.3. Program

//Create an alias

```
LIBNAME TEST "\\\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
%PARAM X_DATE LABEL="Choose a day " NOTNULL TYPE=DATEDEFAULT=TODAY;  
//Print the parameter  
PROC PRINT  
PUT FORMAT(%X_DATE, "d/m/yyyy");
```

## 5.35.16.4. Result

Launch report



## 5.35.17. %PARAM - ??LOOKUP - ??XLOOKUPCODE

### 5.35.17.1. Goal

Use interactive user input to select one or more items of a list referenced in a reference table.

## 5.35.17.2. Features being illustrated

- `%PARAM` statement
- `%Param` - Informats

## 5.35.17.3. Program

```
/*CODES_1 returns the code of the selected option in a standard reference table*/
```

```
%PARAM CODES_1 TYPE = STRING LABEL = "Standard  
instrument categories{F}Catégories d'instrument standardes" INFORMAT="??  
LOOKUPCODE(VCINVESTINSCLAS);
```

```
/*CODES_2 returns the labels of the selected options in a standard reference table*/
```

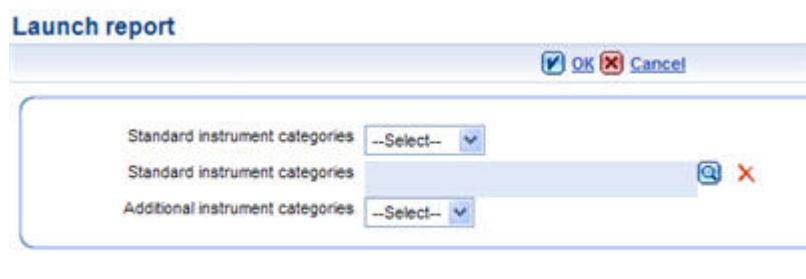
```
%PARAM CODES_2 TYPE = STRING LABEL = "Standard  
instrument categories{F}Catégories d'instrument standardes" INFORMAT="??  
XLOOKUP(VCINVESTINSCLAS);
```

```
/*CODES_3 returns the label of the selected option in a user defined options list that is  
associated to the additional field USERTEXT16*/
```

```
%PARAM CODES_3 TYPE = STRING LABEL = "Additional  
instrument categories{F}Catégories d'instrument additionnelles" INFORMAT="??  
LOOKUP(VCINVESTINSCLAS.USERTEXT16);
```

## 5.35.17.4. Result

[Launch report](#)



## 5.35.18. %PARAM - ??PICKID(SQL- query) - ??FILTER

### 5.35.18.1. Goal

Use interactive user input to select an item of a database table, with the list of choices being filtered according to the **user region**.

### 5.35.18.2. Features being illustrated

- [%PARAM statement](#)
- [%Param - Informats](#)

### 5.35.18.3. Program

```
/*display the list of all the funds in the database for user choice*/
```

```
%PARAM FUNDNAME_1 LABEL = "All the funds of the database" INFORMAT="???
PICKID(SELECT A.IQID, A.FUND FROM VCFUND A WHERE (A.FSTATUS=0))"
NOTNULL;
```

```
//display the list of funds being part of the user region for user choice
```

```
%PARAM FUNDNAME_2 LABEL = "All the funds of user region" INFORMAT="??
PICKID(SELECT A.IQID, A.FUND FROM VCFUND A WHERE (??FILTER) AND
(A.FSTATUS=0))" NOTNULL;
```

## 5.35.19. %PARAM - ??PICKID\_{VAR}

### 5.35.19.1. Goal

Link two %PARAM statements, and use the value(s) returned by the first one to define the data selection to be presented to the user by the second one.

### 5.35.19.2. Features being illustrated

- [%PARAM statement](#)
- [%Param - Informats](#)

### 5.35.19.3. Program

```
/*Display the list of all the funds of the user region for user choice*/
```

```
%PARAM FUNDS LABEL = "1. Select funds{F}1. Sélectionnez les fonds"
INFORMAT="??XPICKID(SELECT A.IQID, A.FUND FROM VCFUND A WHERE (???
FILTER) AND (A.FSTATUS=0))" NOTNULL;

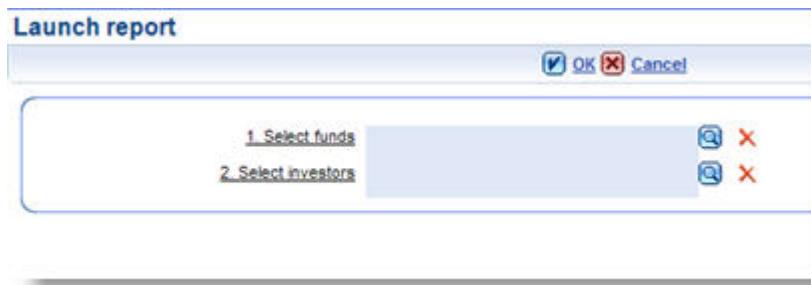
/*According to the funds being selected, Display the list of investors*/

%PARAM INVESTORS LABEL = "2. Select investors{F}2. Sélectionnez les
investisseurs" INFORMAT="??XPICKID(SELECT A.IQID,A.LIBELLE +'-'+F.FUND FROM
VCSUBSCRIBER A, VCFUND F WHERE (A.FUND=F.IQID) AND (A.FUND in(?));
{FUNDS})" NOTNULL;
```

#### 5.35.19.4. Comment

- Notice the way the first macro variable is referenced from within the second %PARAM statement: the SQL query signals the existence of a macro variable: %PARAM INVESTORS ... (SELECT ... (A.FUND in (?));...). The macro variable is delivered after the semicolon following the SQL query: %PARAM INVESTORS ... (SELECT ... (A.FUND in (?));{FUNDS}). The (?) can be anywhere in the SQL query, but the macro variable reference must always follow the semicolon after the SQL query.
- The SQL query is specific of Microsoft SQL Server, because it uses the operator of concatenation '-' (... A.LIBELLE +'-'+F.FUND ...).

#### 5.35.19.5. Result



#### 5.35.20. %PARAM - ??XPICKID - (?) - (?)

##### 5.35.20.1. Goal

Link two %PARAM statements, and use the value(s) returned by the first one to define the data selection to be presented to the user by the second one.

## 5.35.20.2. Features being illustrated

- %PARAM statement
- %Param - Informats

## 5.35.20.3. Program

```
/*Display the list where users choose to run a report either for the 'Managed Fund' or  
'SPV'*/
```

```
%PARAM REPORT_LEVEL LABEL = "Report Level" INFORMAT="??  
CHOICE(Managed Fund, SPV);
```

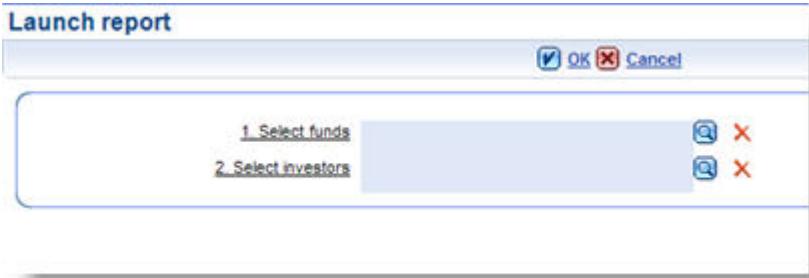
```
/*According to previous selection, two different lists display: funds of status 7 for 'SPV'  
and funds of status 0 for 'Managed funds'*/
```

```
%PARAM MANAGED_FUND_IQID LABEL ="Fund:" INFORMAT="??XPICKID(SELECT  
IQID, FUND FROM VCFUND WHERE (FSTATUS = 7 AND (?) = 'SPV') OR (FSTATUS =  
0 AND (?) = ('Managed Fund');{REPORT_LEVEL};{REPORT_LEVEL});
```

## 5.35.20.4. Comment

- Notice the way the first macro variable is referenced from within the second %PARAM statement: the SQL query signals the existence of a macro variable: %PARAM Managed\_FUND\_IQID... (SELECT ... WHERE(FSTATUS = 7 AND (?) ='SPV')). The macro variable is delivered after the semicolon following the SQL query: %PARAM Managed\_FUND\_IQID ... (SELECT ...;{REPORT\_LEVEL}). As there are two conditions, and two references to the previous variable, you need to repeat the referenced variable at the end: {REPORT\_LEVEL};{REPORT\_LEVEL}
- The (?) can be anywhere in the SQL query, but the macro variable reference must always follow the semicolon after the SQL query.

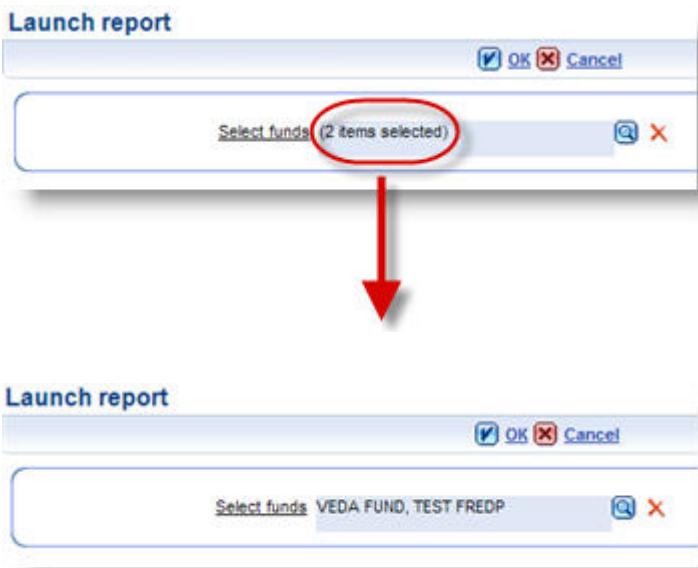
## 5.35.20.5. Result



## 5.35.21. %PARAM - ??XPICKID - SHOWSELECTEDITEMS

### 5.35.21.1. Goal

When using interactive user input, display the items being selected to the user, instead of a "" comment:



### 5.35.21.2. Features being illustrated

- %PARAM statement
- %Param - Informats

### 5.35.21.3. Program

```
/*Display the list of all the funds of the user region for user choice*/  
  
%PARAM FUNDS LABEL = "Select funds{F}Sélectionnez les fonds"  
SHOWSELECTEDITEMS INFORMAT="??XPICKID(SELECT A.IQID, A.FUND FROM  
VCFUND A WHERE (??FILTER) AND (A.FSTATUS=0))" NOTNULL;
```

### 5.35.21.4. Comment

- You must use SHOWSELECTEDITEMS in combination with multi-items selection informats, such as ??XID and ??XPICKID.

## 5.35.22. %PARAM - ??XPICKID with SQL query

### 5.35.22.1. Goal

Filter instruments according to instrument class and label.

### 5.35.22.2. Features being illustrated

- %PARAM statement
- %Param - Informats

### 5.35.22.3. Program

```
/*Filter security types according to instrument class and code*/
```

```
%PARAM SECURITY_TYPES LABEL= "Instruments" TYPE=STRINGINFORMAT="??  
XPICKID(SELECT CODE, LIBELLE FROM REINSTRUMENTTYPE A WHERE CLASS1  
IN(2,3,4) AND CODE IN('PP','CC','ACC'))";
```

## 5.36. Examples - functions and operators

The following examples illustrate functions:

- DATEADD() - FORMAT()
- DATEDIFF() - FORMAT()
- DATEDIFF() - SEMESTERENDDATE() - 1
- DATEDIFF() - SEMESTERENDDATE() - 2
- DATEDIFF() - QuarterBegDate() - DateADD()
- DMY() - MDY() - FORMAT()
- FILENAME() - FILE()
- GETTEMPPATH()
- GETUSERID()\_GETUSERINFO()
- LINKFILES() - LINKFILE() - FILENAME()
- LINKFILES() - LINKFILE() - FILENAME() - %PARAM
- LINKFILES() - LINKFILE() - PROC MEMORY - Macro controls
- LOOKUP()\_LOOKUPCODE()
- ROUND()
- WEEKDAY()
- YMD()
- YMD() - FORMAT()

The following examples illustrate operators:

- PROC SORT - (WHERE - NOT...IN)

### 5.36.1. DATEADD() - FORMAT()

#### 5.36.1.1. Goal

Add a specified period to a given date.

#### 5.36.1.2. Features being illustrated

- Functions - date

### 5.36.1.3. Program

```
//Create the alias that points the existing table location

LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";

/*Create the table that receives date formats*/

DATA TEST.ADDED_DATES;

COLUMN X_DATE TYPE=DATE LABEL="Today";

COLUMN X_DATE_DAY LABEL="Today + 1 DAY" WIDTH=160;

COLUMN X_DATE_MONTH LABEL="Today + 2 MONTHS" WIDTH=160;

COLUMN X_DATE_YEAR LABEL="Today + 10 YEARS" WIDTH=160;

X_DATE = TODAY();

X_DATE_DAY = FORMAT(DATEADD("DAY",1,X_DATE), "d/M/yyyy");

X_DATE_MONTH = FORMAT(DATEADD("MONTH",2,X_DATE), "d/M/yyyy");

X_DATE_YEAR = FORMAT(DATEADD("YEAR",10,X_DATE), "d/M/yyyy");

OUTPUT;

RUN;

//Display the result

PROC PRINT DATA=TEST.DATE_FORMATS LABEL NOOBS;

RUN;
```

### 5.36.1.4. Result

Today	Today + 1 DAY	Today + 2 MONTHS	Today + 10 YEARS
06/25/2007	26/6/2007	25/8/2007	25/6/2017

## 5.36.2. DATEDIFF() - FORMAT()

### 5.36.2.1. Goal

Return the difference between two dates.

### 5.36.2.2. Features being illustrated

- Functions - date

### 5.36.2.3. Program

```
//Create the alias that points the existing table location  
  
LIBNAME TEST "\{\Private\}\TESTEXAMPLES\TablesFormation";  
  
/*Create the table that receives dates and difference*/  
  
DATA TEST.DATES_DIFF;  
  
COLUMN X_DATE_1 LABEL="Date 1" TYPE="DATE";  
  
COLUMN X_DATE_2 LABEL="Date 2" TYPE="DATE";  
  
/* if you don't specify the type of the column, eFront Script interprets the column type as  
of being 'String', and not of 'Date'. DATEDIFF() then re-interprets the strings 'X_DATE_1'  
and X_DATE_2 to convert them into dates. According to the local settings, the strings  
might be interpreted differently. For example '01/06/2007' can be interpreted as 'June, 1,  
2007' or as 'January, 6, 2007'*/  
  
COLUMN X_DATE_DIFF_DAY LABEL="Difference in days" WIDTH=150;  
  
COLUMN X_DATE_DIFF_MONTH LABEL="Difference in months" WIDTH=150;  
  
COLUMN X_DATE_DIFF_YEAR LABEL="Difference in years" WIDTH=150;  
  
/*Note that you need to use an upper-case m (M) for the month. If a lower-case m is  
used, the program will return '0' for the month.*/  
  
X_DATE_1 = FORMAT(TODAY(),"d/M/yyyy");  
  
X_DATE_2 = FORMAT(DATEADD("MONTH",10,X_DATE_1), "d/M/yyyy");
```

```

X_DATE_DIFF_DAY = DATEDIFF("DAY",X_DATE_1,X_DATE_2);
X_DATE_DIFF_MONTH = DATEDIFF("MONTH",X_DATE_1,X_DATE_2);
X_DATE_DIFF_YEAR = DATEDIFF("YEAR",X_DATE_1,X_DATE_2);
OUTPUT;
RUN;

PROC PRINT DATA = TEST.DATES_DIFF LABEL NOOBS;
RUN;

```

### 5.36.2.4. Result

Date 1	Date 2	Difference in days	Difference in months	Difference in years
25/6/2007	25/4/2008	305	10	1

## 5.36.3. DATEDIFF() - SEMESTERENDDATE() - 1

### 5.36.3.1. Goal

Count the number of semesters between two dates.

### 5.36.3.2. Features being illustrated

- Functions - date

### 5.36.3.3. Program

//Create the alias to the folder to work in

```
LIBNAME TEST "\{Private}\TEST\EXAMPLES\TablesFormation";
```

```
//Create the table that receives dates and difference*/
```

```
DATA TEST.DATES_DIFF;
```

```
COLUMN X_DATE_1 LABEL="Date 1" TYPE="DATE";
```

```
COLUMN X_DATE_2 LABEL="Date 2" TYPE="DATE";
```

```
/* if you don't specify the type of the column, eFront Script interprets the column type as  
of being 'String', and not of 'Date'. DATEDIFF() then re-interprets the strings 'X_DATE_1  
and X_DATE_2 to convert them into dates. According to the local settings, the strings  
might be interpreted differently. For example '01/06/2007' can be interpreted as 'June, 1,  
2007' or as 'January, 6, 2007'*/
```

```
COLUMN X_DATE_NB_SEMESTER LABEL="Number of semesters";
```

```
//Note that you need to use an upper-case m (M) for the month. If a lower-case m is  
used, the program will return '0' for the month.
```

```
X_DATE_1 = FORMAT(YMD (2004,12,31),"dd/MM/yyyy");
```

```
X_DATE_2 = FORMAT(SEMESTERENDDATE(TODAY()),"dd/MM/yyyy");
```

```
X_DATE_NB_SEMESTER = DATEDIFF("SEMESTER",X_DATE_1,X_DATE_2);
```

```
OUTPUT;
```

```
RUN;
```

```
PROC PRINT DATA = TEST.DATES_DIFF LABEL NOOBS;
```

```
RUN;
```

#### 5.36.3.4. Result

Date 1	Date 2	Number of semesters
31/12/2004	30/06/2008	7

#### 5.36.4. DATEDIFF() - SEMESTERENDDATE() - 2

##### 5.36.4.1. Goal

Test a series of instructions using DATEDIFF().

### 5.36.4.2. Features being illustrated

- Functions - date
- PUT statement

### 5.36.4.3. Program

//Test and write some examples.

PROC PRINT;

```
PUT "Closing date of the current semester: "
FORMAT(SEMESTERENDDATE(TODAY()), 'd/MM/yy');
```

```
PUT CRLF();
```

```
PUT "Number of semesters between 31/12/2004 and the closing date of the current
semester:"
```

```
DATEDIFF("SEMESTER","31/12/2004",SEMESTERENDDATE(TODAY()));
```

```
PUT CRLF();
```

```
PUT "Number of semesters between 31/12/2004 and the closing date of the current
semester + 1:"
```

```
DATEDIFF("SEMESTER","31/12/2004",DATEADD("DAY",1,SEMESTERENDDATE(TOD
AY())));
```

```
RUN;
```

### 5.36.4.4. Result

```
Closing date of the current semester:
30/06/08

Number of semesters between 31/12/2004 and the closing date of the current semester:
7

Number of semesters between 31/12/2004 and the closing date of the current semester + 1:
8
```

## 5.36.5. DATEDIFF() - QuarterBegDate() - DateADD()

### 5.36.5.1. Goal

Create a table of quarterly positions.

### 5.36.5.2. Features being illustrated

- Functions - date
- %DO WHILE...%LOOP WHILE
- MIN statement

### 5.36.5.3. Program

//Declare macro variables

```
%Include "FV_For_Pages";  
%Let Date = Today();  
%Let Min_Date = "01012007"d;  
/*  
Proc Means Data = Work.Transactions Out = Work.Min_Date;  
Min Reference_Date (Name = Min_Date);  
Run;  
Data Work.Min_Date;  
Set Work.Min_Date;  
%Let Min_Date = Min_Date;  
Run;  
*/  
Data Work.Quarters;
```

```
Column Year Type = "INTEGER";
Column Quarter Type ="INTEGER";
Column Quarter_Number Type = "INTEGER";
Column Year_Number Type = "INTEGER";
Column Quarter_Begin_Date Type = "DATE";
Do While DateDiff("QUARTER", %Date, %Min_Date) <= 0
Year = Year(%Min_Date);
Quarter = Quarter(%Min_Date);
Quarter_Number = Datediff("QUARTER", %Date, %Min_Date);
Year_Number = DateDiff("YEAR", %Date, %Min_Date);
Quarter_Begin_Date = QuarterBegDate(%Min_Date);
%Let Min_Date = DateAdd("QUARTER", 1, %Min_Date);
Output;
Loop;
Run;
```

### 5.36.5.4. Result

	YEAR	QUARTER	QUARTER_NUMBER	YEAR_NUMBER	QUARTER_BEGIN_DATE
1	2007	1	-11	-2	01/01/2007
2	2007	2	-10	-2	01/04/2007
3	2007	3	-9	-2	01/07/2007
4	2007	4	-8	-2	01/10/2007
5	2008	1	-7	-1	01/01/2008
6	2008	2	-6	-1	01/04/2008
7	2008	3	-5	-1	01/07/2008
8	2008	4	-4	-1	01/10/2008
9	2009	1	-3	0	01/01/2009
10	2009	2	-2	0	01/04/2009
11	2009	3	-1	0	01/07/2009
12	2009	4	0	0	01/10/2009

### 5.36.5.5. Comments

The entire set of columns coming together with the quarterly positions is very useful when filtering the tables in reports/documents.

## 5.36.6. DMY() - MDY() - FORMAT()

### 5.36.6.1. Goal

Display a date.

### 5.36.6.2. Features being illustrated

- Functions - date

### 5.36.6.3. Program

//Create the alias that points the existing table location

```
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";
```

```
/*Create the table that receives dates and values*/
```

```
DATA TEST.WRITING_DATES;
```

```
COLUMN X_DATE_DAY LABEL="Day";
```

```
COLUMN X_DATE_MONTH LABEL="Month";  
COLUMN X_DATE_YEAR LABEL="Year";  
COLUMN X_DATE_DMY LABEL="DMY";  
COLUMN X_DATE_MDY LABEL="MDY";  
X_DATE_DAY = DAY(TODAY());  
X_DATE_MONTH = MONTH(TODAY());  
X_DATE_YEAR = YEAR(TODAY());  
X_DATE_DMY = FORMAT(DMY(X_DATE_DAY,X_DATE_MONTH,X_DATE_YEAR),  
"d/M/yyyy");  
X_DATE_MDY = FORMAT(MDY(X_DATE_MONTH,X_DATE_DAY,X_DATE_YEAR),  
"M/d/yyyy");  
OUTPUT;  
RUN;  
PROC PRINT DATA = TEST.WRITING_DATES LABEL NOOBS;  
RUN;
```

#### 5.36.6.4. Result

Day	Month	Year	DMY	MDY
26	7	2019	26/7/2019	7/26/2019

#### 5.36.7. FILENAME() - FILE()

##### 5.36.7.1. Goal

Insert images that have been uploaded to the application's database into an EXCEL report.

### 5.36.7.2. Features being illustrated

- Functions - files and folders
- IF...THEN...ELSE control
- EF\_IMG()

### 5.36.7.3. Program

```
//Build the table that receives the uploaded image file

DATA WORK.report;

COLUMN I TYPE=INTEGER;

COLUMN LOGO WIDTH=450;

//Search for the file CMYKlogo.jpg

FOR I = 1 TO FILES("\Shared region(company)\VAL")

IF UPCASE(FILENAME(I)) = "CMYKLOGO.JPG" THEN

LOGO = FILE(I);

END;

NEXT;

// If the picture we are looking for doesn't exist then take the file empty.jpg

IF LEN(LOGO) = 0 THEN

FOR I = 1 TO FILES("\Shared region(company)\VAL")

IF UPCASE(FILENAME(I)) = "EMPTY.JPG" THEN

LOGO = FILE(I);

END;

NEXT;
```

```
END;  
OUTPUT;  
RUN;  
PROC PRINT DATA=WORK.REPORT;  
RUN;
```

#### 5.36.7.4. Comment

- The function FILES() only counts the files of the folder being identified, underlying folders are not taken into account.
- Use the EF\_IMG() macro to insert the image into the EXCEL report, ex:  
EF\_IMG("LOGO";C5:C6;"27";;;100")

#### 5.36.7.5. Result

I	LOGO
1	5 C:\Inetpub\wwwroot\f\61s\website\download\39b950e1429044f4b3415 D14E896283824074A8E2C1DB5F554F55\CMYKlogo.jpg

### 5.36.8. GETTEMPPATH()

#### 5.36.8.1. Goal

Create a CSV file on the server (from an eFront Report table, and display a link in HTML output to download this file.

#### 5.36.8.2. Features being illustrated

- Functions - special
- PROC EXPORT step
- %LET statement

#### 5.36.8.3. Program

//Build a temporary table

```
DATA WORK.TMP;  
COLUMN NB TYPE=FLOAT;  
COLUMN NAME TYPE=STRING;  
//Instantiate the table  
NB = 1.3456;  
NAME = "xdrdgj";  
OUTPUT;  
RUN;  
PROC PRINT DATA=WORK.TMP;  
RUN;  
//Create a macro-variable for access path to the CSV table  
%LETFILE = GETTEMPPATH() & "\test.csv";  
//Export the temporary data table and create an CSV file on the server  
PROC EXPORT DATA=WORK.TMP DROP NOERROR FILE=%FILE NOHEADER  
SEMICOLUMN;  
RUN;
```

#### 5.36.8.4. Comment

- Through the use of PROC EXPORT, the application generates a .CSV file from the temporary data table. Semicolons are used as separators. The export file doesn't integrate the column header from the source data table.

## 5.36.9. GETUSERID()\_GETUSERINFO()

### 5.36.9.1. Goal

Retrieve information about the current user of the application.

### 5.36.9.2. Features being illustrated

- Functions - special

### 5.36.9.3. Program

```
// USERNUMBER => Unique number that is assigned to the current user

%LET USERID_1 = GetUserID();

// UserID => Login

%LET USERID = GetUserInfo(GetUserID(),"USERID1");

// User => First Name and Last Name

%LET USER = GetUserInfo(GetUserID(),"FIRSTNAME1")& " " &
GetUserInfo(GetUserID(),"LASTNAME1");

//User => Country

%LET USER_COUNTRY = GetUserInfo(GETUSERID(),"COUNTRY1");

PROC PRINT;

PUT "Userid_1: " & %USERID_1;

PUT "Userid: " & %USERID;

PUT "User: " & %USER;

PUT "User country: " & %USER_COUNTRY;

RUN;
```

### 5.36.9.4. Available fields

List of meaningful fields that can be passed as <field\_name>:

- **IQID1** - User IQID
- **USERID1** - User ID
- **LASTNAME1** - User Lastname
- **FIRSTNAME1** - User Firstname
- **MAILID1** - First part of mail address (before @)
- **MAILNAME1** - User mail name
- **USERPROFILE** - IQID of current user profile
- **USERLIBPROFILE** - Current profile label
- **EMAIL1** - User full email
- **ACCOUNT1** - IQID of user account
- **PHONE1** - User phone number
- **MOBILE1** - User mobile phone number
- **ADDRESS11** - User address 1

### 5.36.9.5. Comment

- In EXCEL, use =EF\_MACRO("USER") to print the user.

### 5.36.9.6. Result

```
Userid_1: 8C3D53F235E542EF9930DDE763B3A283
Userid: FVBP.1
User: Monika Duvinage
User country: FRA
```

## 5.36.10. LINKFILES() - LINKFILE() - FILENAME()

### 5.36.10.1. Goal

List all the files being attached to a fund.

### 5.36.10.2. Features being illustrated

- Functions - files and folders

- %PARAM statement

### 5.36.10.3. Program

```
//Build the table that receives the uploaded image file  
  
%Param ID Informat="??PICKID(VCFUND;FUND)";  
  
%FOR %I = 1 %TO (LINKFILES(%ID))  
  
TraceFILENAME(%I) &" - "& LINKFILE(%I);  
  
%NEXT;
```

### 5.36.10.4. Comment

- You may want to use this program to pull out the fund logos that have been uploaded for a fund.

## 5.36.11. LINKFILES() - LINKFILE() - FILENAME() - %PARAM

### 5.36.11.1. Goal

Launch from an EXCEL report files that are linked to a fund as URLs.

### 5.36.11.2. Features being illustrated

- Functions - files and folders
- %PARAM statement

### 5.36.11.3. Program

```
//Build the table that receives the uploaded image file  
  
%Param ID Informat="??PICKID(VCFUND;FUND)";  
  
// Address of the website  
  
%Let WEBSITE_URL = "http://cordiant-dev.na.frontsrv.com";  
  
// Path given by LINKFILE function
```

```
%Let PATH_TO_CHANGE = "C:\websites\cordiant-de\website";  
%FOR %I = 1 %TO (LINKFILES(%ID))  
//Proc print; Put LINKFILE(%I); Run; // Uncomment this line to see the complete path.  
Proc print;  
  
Put "<a href="" &  
Replace(Replace(LINKFILE(%I),%PATH_TO_CHANGE,%WEBSITE_URL),"\","/") &  
""">" & FILENAME(%I) & "</a>";  
  
Run;  
%NEXT;
```

## 5.36.12. LINKFILES() - LINKFILE() - PROC MEMORY - Macro controls

### 5.36.12.1. Goal

Get for each contact the image file and integrate it into an EXCEL report.

### 5.36.12.2. Features being illustrated

- Functions - files and folders
- PROC MEMORY step
- %IF...%THEN...%END
- Flags

### 5.36.12.3. Program

```
LibName VIEWS 'VIEWS';
```

```
LibName OUTPUT 'OUTPUT';
```

```
%PARAM AS_OF_DATE LABEL= "<b><font color='#000000'>Date</font></b>" Type =  
Date Default = Today NOTNULL;
```

```
%PARAM COMPANY_ID LABEL= "<b><font color='#000000'>Select Company:</font></b><BR>(Run for all Companies, if left empty) " INFORMAT= "??
```

```
XPICKID(SELECT A.IQID, A.NAME FROM SFAACCOUNT A WHERE ??FILTER AND
A.STATUS=1 Order By A.Name)";

%Let List = %COMPANY_ID;

%Include "Empty List";

%Undefine List;

DATA Work.T_Contact_Info;

//SET Views.V_Contact;

SET Views.V_DIRECTORS;

Column Contact_Name Type = String;

Contact_Name = Contact_First_Name + " " + Contact_Last_Name;

RUN;

// ***** Get contact Image ***** //

DATA Work.TEMP(Keep = CONTACT_PICTURE CONTACT_ID);

SET Work.T_Contact_Info;

COLUMN CONTACT_PICTURE WIDTH=340;

COLUMN I TYPE = INTEGER;

FOR I=1 TO (LINKFILES(CONTACT_ID)) //LINKFILESEXT(CONTACT_ID,".PNG;.JPG")

CONTACT_PICTURE = LINKFILE(I);

OUTPUT;

NEXT;

RUN;
```

```
// *****
DATA Work.T_Contact_Info;
MERGE Work.T_Contact_Info (IN = OK)
Work.TEMP;
BY CONTACT_ID;
_OUTPUT_=OK;
RUN;
DATA OUTPUT.T_Contact_Info;
SET Work.T_Contact_Info;
RUN;
// ***** Contact Notes *****
%DEFINE Filter_Note_Date;
%LET Note_Date = %As_Of_Date;
%INCLUDE "P_Contact_Notes";
%UNDEFINE Filter_Note_Date;
%UNDEFINE Note_Date;
// *****
%INCLUDE "P_Contact_Directorships";
%INCLUDE "P_Contact_Education";
%IF (%Filled_Collection <> 0 ) %THEN
DATA Work.T_Contact_Filter_By_Directorship_company( WHERE COMPANY_ID IN
(%COMPANY_ID) );
```

```
SET Work.T_Contact_Directorships;  
RUN;  
  
PROC MEANS DATA = Work.T_Contact_Filter_By_Directorship_company;  
CLASS CONTACT_ID;  
RUN;  
  
DATA Work.T_CONTACT_INFO;  
  
MERGE Work.T_Contact_Filter_By_Directorship_company (IN = OK)  
Work.T_CONTACT_INFO;  
BY CONTACT_ID;  
_OUTPUT_ = OK;  
RUN;  
  
PROC MEMORY UNLOAD DATA = Work.T_Contact_Filter_By_Directorship_company;  
RUN;  
  
%END;
```

## 5.36.12.4. Result

N15

	A	B	C	D	E	F	G	H	I	J	K	L																				
1																																
2																																
3																																
4																																
5	 <p><b>Mahmood Al-Kooheji</b></p> <p>Bahrain Mumtalakat Holding Co.</p>																															
6																																
7																																
8																																
9	Office:	+973 1756 1111																														
10	Mobile:	+973 38994777																														
11	Fax:																															
12	Email:	mahmood.alkooheji@bmhc.bh																														
13																																
14	<b>Profile</b>	<p>Mahmood Hashim Al Kooheji is the Chief Executive Officer of Bahrain Mumtalakat Holding Company ("Mumtalakat"), the investment arm of the Kingdom of Bahrain. He is responsible for devising and executing its strategy to manage and grow the Kingdom's strategic non-oil and gas related assets.</p> <p>As the CEO of Mumtalakat, Mr. Al Kooheji oversees the management of the company's stakes in over 35 commercial enterprises, and the growth of its portfolio valued at approximately BD2.7 billion (US\$7.1 billion) as of June 30, 2012. He also directs the company's initiatives to firmly implement the highest standards of transparency and corporate governance across Mumtalakat and its diverse holdings.</p>																														
15	<b>Board Membersh</b>	<p>He is the Chairman of Alba, and sits on the board of Gulf Air. He is also a board member at McLaren Automotive Limited and McLaren Group Limited. In addition, he sits on the boards of the Crown Prince's International Scholarship Program ("CPISP"), Durra Khaleej Al Bahrain Company, the Arab Petroleum Investment Corporation ("APICORP"), Governors of the Royal College of Surgeons in Ireland, Bahrain.</p>																														
16																																
17	<p style="text-align: center;"><b>Education background</b></p> <table border="1"> <thead> <tr> <th>Date</th> <th>Level</th> <th>Course details</th> <th>University</th> </tr> </thead> <tbody> <tr> <td></td> <td>MBA</td> <td>Business Administration</td> <td>Brunel University</td> </tr> <tr> <td></td> <td>B.Sc.</td> <td>Mechanical Engineering</td> <td>Staffordshire University</td> </tr> </tbody> </table>												Date	Level	Course details	University		MBA	Business Administration	Brunel University		B.Sc.	Mechanical Engineering	Staffordshire University								
Date	Level	Course details	University																													
	MBA	Business Administration	Brunel University																													
	B.Sc.	Mechanical Engineering	Staffordshire University																													
18																																
19																																
20																																
21																																
22	<p style="text-align: center;"><b>Directorships</b></p> <table border="1"> <thead> <tr> <th>Company</th> <th>Representing</th> <th>Appointment Date</th> <th>Renewal Date</th> </tr> </thead> <tbody> <tr> <td>Arab Petroleum Investment Cor</td> <td>Bahrain Mumtalakat Holding Co.</td> <td></td> <td>01-04-11</td> </tr> <tr> <td>Falcon Group Holding Compan</td> <td>Bahrain Mumtalakat Holding Co.</td> <td></td> <td></td> </tr> <tr> <td>Hawar Holding Compan</td> <td>Bahrain Mumtalakat Holding Co.</td> <td>01-01-06</td> <td>01-01-07</td> </tr> <tr> <td>Gulf Air Group Holding</td> <td>Bahrain Mumtalakat Holding Co.</td> <td></td> <td></td> </tr> </tbody> </table>												Company	Representing	Appointment Date	Renewal Date	Arab Petroleum Investment Cor	Bahrain Mumtalakat Holding Co.		01-04-11	Falcon Group Holding Compan	Bahrain Mumtalakat Holding Co.			Hawar Holding Compan	Bahrain Mumtalakat Holding Co.	01-01-06	01-01-07	Gulf Air Group Holding	Bahrain Mumtalakat Holding Co.		
Company	Representing	Appointment Date	Renewal Date																													
Arab Petroleum Investment Cor	Bahrain Mumtalakat Holding Co.		01-04-11																													
Falcon Group Holding Compan	Bahrain Mumtalakat Holding Co.																															
Hawar Holding Compan	Bahrain Mumtalakat Holding Co.	01-01-06	01-01-07																													
Gulf Air Group Holding	Bahrain Mumtalakat Holding Co.																															
23																																
24																																
25																																
26																																
27																																
28																																
29																																
30																																

Mahmood Al-Kooheji(1) Report Template



You get one EXCEL workbook sheet per contact.

## 5.36.12.5. EXCEL Report Template

 To integrate the picture in the EXCEL workbook sheet, the EF\_IMG() macro is used.

## 5.36.13. LOOKUP()\_LOOKUPCODE()

### 5.36.13.1. Goal

Get the code and the label of options being part of the options list which is associated with an additional field

### 5.36.13.2. Features being illustrated

- Functions - special

### 5.36.13.3. Prerequisite

The LOOKUP() and LOOKUPCODE() functions apply specifically to the list of options being available for additional fields. Therefore you can only use this function if additional fields have been defined, and if a list of options has been defined for the additional field you refer to, such as shown below:

1. Add an additional field

**Table**

Sauver Supprimer Annuler

**e Champs additionnels**

Table VC/Instrument/Investissement

Description

	Champ Utilisateur	Section	Libellé	Format	Lien vers
	Texte[01]		Nominal	Compact	--Choix--
	Texte[02]		Durée	Compact	--Choix--
	Texte[03]		Marge cash	Compact	--Choix--
	Texte[04]		Call protection	Compact	--Choix--
	Texte[05]		Nantissement (rang)	Compact	--Choix--
	Texte[06]		N° cotation	Compact	--Choix--
	Texte[07]		Montant résiduel à émettre	Compact	--Choix--
	Texte[08]		Quotité à l'origine	Compact	--Choix--
	Texte[09]		Echangeable en quotités après opération	Compact	--Choix--
	Texte[10]		Prix d'exercice	Compact	--Choix--
	Texte[11]		Dilution totale en % capital	Compact	--Choix--
	Texte[12]		Dilution variable (oui/non)	Compact	--Choix--
	Texte[13]		Critères de variabilité	Compact	--Choix--
	Texte[14]		Clause anti-dilution	Compact	--Choix--
	Texte[15]		Clause particulières	Compact	--Choix--
	Numérique[01]		Prix exercice BSA	--Choix--	--Choix--
	Texte[16]		Catégorie d'instrument	--Choix--	--Choix--
	...Choix...			--Choix--	--Choix--

- Define the list of options for the additional field

**Table de valeurs pour le champ additionnel**

Sauver

**Aide**

Vous pouvez associer une table de référence à votre champ additionnel. Pour cela, rentrez dans cet écran la liste des codes inter correspondant.  
Pour associer votre champ additionnel à une table de référence existante, vous devez ne saisir qu'une seule ligne avec un code s utilisables.

- @@USERS : associe votre champs additionnel à la liste des utilisateurs
- @@INUSERS : associe votre champs additionnel à la liste des utilisateurs actifs (non lockés)
- @@FOLDERS : associe votre champs additionnel à la liste des dossiers
- @@REFTABLE(nomtable) : associe votre champs additionnel à la table de référence spécifiée
- @@CATEGORY(type) : Liste des catégories d'un certain type (passer en paramètre le code ex : EVCA, AFIC...)

**Saisie**

Code	Libellé
0	Non renseigné
1	Equity
2	Quasi equity
3	Vendor's loan
4	Junior mezzanine
5	Senior warranteed mezzanine
6	Senior unwarranteed mezzanine
7	Senior debt
8	Other debt

### 5.36.13.4. Program

//Define the micro variables that held the reference to a particular option being associated to the additional field

```
%LET InstrLabel1 = LOOKUP("VCINVESTMENTINS.USERTEXT16", 1);
%LET InstrCode1 = LOOKUPCODE("VCINVESTMENTINS.USERTEXT16", "Equity");
%LET InstrLabel2 = LOOKUP("VCINVESTMENTINS.USERTEXT16", 2);
%LET InstrCode2 = LOOKUPCODE("VCINVESTMENTINS.USERTEXT16", "Quasi equity");
```

PROC PRINT;

```
PUT %InstrLabel1;
```

```
PUT %InstrCode2;
```

```
PUT %InstrLabel1;
```

```
PUT %InstrCode2;
```

```
RUN;
```

### 5.36.13.5. Result

Equity1

Quasi equity2

## 5.36.14. PROC SORT - (WHERE - NOT...IN)

### 5.36.14.1. Goal

Sort a table using an exclusive condition.

### 5.36.14.2. Features being illustrated

- Operators

### 5.36.14.3. Program

//Display the result

```
PROC SORT DATA=T_PORTFOLIO (WHERE NOT COMP IN("Cash & other assets","Carry","Leverage")) OUT=T_TO_HOLDINGS;
```

```
BY DESCENDING XCURVALUATION;
```

```
RUN;
```

## 5.36.15. ROUND()\_CINT()

### 5.36.15.1. Goal

Two methods of round numbers

## 5.36.15.2. Features being illustrated

- Functions - numbers

### 5.36.15.3. Program

```
//Create the alias that points the existing table location  
LIBNAME TEST "\\\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
  
/*Create the table that receives date formats*/  
  
DATA TEST.ROUNDED_VALUE;  
  
COLUMN MyVALUE TYPE=FLOAT;  
  
COLUMN UseRound;  
  
COLUMN UseInteger;  
  
Myvalue = 235.21573;  
  
UseRound = ROUND(MyValue,2);  
  
UseInteger = CINT(MyValue);  
  
OUTPUT;  
  
Myvalue = 1.4;  
  
UseRound = ROUND(MyValue,0);  
  
UseInteger = CINT(MyValue);  
  
OUTPUT;  
  
Myvalue = 1.6;  
  
UseRound = ROUND(MyValue,0);  
  
UseInteger = CINT(MyValue);
```

```
OUTPUT;  
Myvalue = 1.6;  
UseRound = ROUND(MyValue-0.5,0);  
UseInteger = CINT(MyValue);  
OUTPUT;  
RUN;  
PROC PRINT DATA=TEST.ROUNDED_VALUE;  
RUN;
```

#### 5.36.15.4. Result

	MYVALUE	USEROUND	USEINTEGER
1	235.21573	235.22	235
2	1.4	1	1
3	1.6	2	2
4	1.6	1	2

#### 5.36.16. WEEKDAY()

##### 5.36.16.1. Goal

Calculate the number of business days prior to a certain date (STARTDATE) to trigger an alert?

##### 5.36.16.2. Features being illustrated

- Functions - date
- Functions - numbers

##### 5.36.16.3. Comment

- WEEKDAY function in eFront Script returns **Monday 1**, whereas EXCEL returns **Sunday 1!**

- We use the ceiling function to help work out the remainder (in Excel we can use the MOD function which does not exist in Front Script).

#### 5.36.16.4. Program

```
LIBNAME USER ":";  
  
%PARAM STARTDATE TYPE=DATE LABEL="Starting from " NOTNULL;  
  
%PARAM DAYSTOADD TYPE=INTEGER LABEL="Business days" NOTNULL;  
  
//Code to create the same result as the MOD function in Excel (to get the remainder  
when dividing by 5 since there are 5 business days in a week)  
  
%LET MODDAYSTOADD=%DAYSTOADD-(5*CEILING(%DAYSTOADD/5));  
  
TRACE %MODDAYSTOADD;  
  
//Calculate the correction to move past the weekend if necessary as a result of the  
remainder days (derived above)  
  
%IF (WEEKDAY(%STARTDATE)+%MODDAYSTOADD)>5 %THEN  
  
%LET CORRECTION=2;  
  
%ELSE %LET CORRECTION=0; %END;  
  
// Derive the new date from the original, plus the number of weeks plus the correction  
  
%LET NEWDATE=CDATE(%MODDAYSTOADD+  
%STARTDATE+(7*CEILING(%DAYSTOADD/5))+%CORRECTION);  
  
// Trace the result  
  
TRACE %NEWDATE;  
  
// This might be useful since you can easily use positive or negative "DAYSTOADD" and  
therefore move backwards and forwards at will.
```

## 5.36.16.5. Result - EXCEL report

## 5.36.17. YMD()

### 5.36.17.1. Goal

Convert an integer into a date and vice versa

### 5.36.17.2. Features being illustrated

- Functions - date

### 5.36.17.3. Program

```
//Create the alias that points the existing table location
```

LIBNAME TEST "\EFRONT\_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";

```
/*Create the table that receives the integer*/  
  
DATA TEST.T_test_date;  
  
COLUMN CYEAR TYPE=INTEGER;  
  
CYEAR = "2007";  
  
OUTPUT;  
  
RUN;  
  
/*Convert the integer into a date*/  
  
DATA TEST.T_test_date;  
  
SET TEST.T_test_date;  
  
COLUMN REAL_DATE TYPE=DATE;  
  
REAL_DATE = YMD(CYEAR,12,31);  
  
OUTPUT;  
  
RUN;  
  
//Display the result  
  
PROC PRINT DATA=TEST.T_test_date LABEL NOOBS;  
  
RUN;
```

#### 5.36.17.4. Result

	REAL_DATE
2007	31/12/2007

## 5.36.18. YMD() - FORMAT()

### 5.36.18.1. Goal

Apply different formats to a date.

### 5.36.18.2. Features being illustrated

- Functions - date

### 5.36.18.3. Program

```
//Create the alias that points the existing table location  
  
LIBNAME TEST "\EFRONT_SUP.2(Private)\TEST\EXAMPLES\TablesFormation";  
  
/*Create the table that receives date formats*/  
  
DATA TEST.DATE_FORMATS;  
  
COLUMN X_DATE TYPE=DATE LABEL="Initial Format" WIDTH=140;  
  
COLUMN X_STR LABEL="Day - Month - Year" WIDTH=140;  
  
COLUMN X_STR_US LABEL="Month - Day - Year" WIDTH=140;  
  
X_DATE = ymd(2001, 2, 3); // store the date 2001-02-03  
  
X_STR = format(X_DATE, "d/M/yyyy");  
  
X_STR_US = format(X_DATE, "M/d/yyyy");  
  
OUTPUT;  
  
RUN;  
  
//Display the result  
  
PROC PRINT DATA=TEST.DATE_FORMATS LABEL NOOBS;  
  
RUN;
```

## 5.36.18.4. Result

Initial Format	Day - Month - Year	Month - Day - Year
02/03/2001	3/2/2001	2/3/2001

## 5.37. Examples - performance increase for eFront Cube programs

The following examples illustrate how to increase the performance of eFront Cube programs:

- [PROC SETFILTER](#) - All funds with and without filter

### 5.37.1. PROC SETFILTER - EXTEND - Global Cube and Filtered Session Cube

#### 5.37.1.1. Goal

Build a global cube and store it on the cube server. Build a session cube based on the global cube whose data are automatically filtered when users open the dashboard, the session cube is linked to.

#### 5.37.1.2. Features being illustrated

- [PROC SETFILTER](#) step
- SQL statement

#### 5.37.1.3. 1. Program A - Build the global cube

```
%LET SQL = "SELECT  
IQID,FUND,ABBREVIATION,FTYPE,FNATURE,FSTAGE,FREGION,STARTDATE,CLOS  
INGDATE,TARGETRAISED,MANAGEMENTFEES,COUNTRY,CURRENCY1,DESCRIP  
TION FROM VCFUND WHERE IQDELETED=0";
```

```
LIBNAME USER ":";
```

```
DATA ALLFUNDS;
```

```
SQL %SQL;
```

```
RUN;
```

```
PROC PRINT DATA=ALLFUNDS;
```

```
RUN;
```

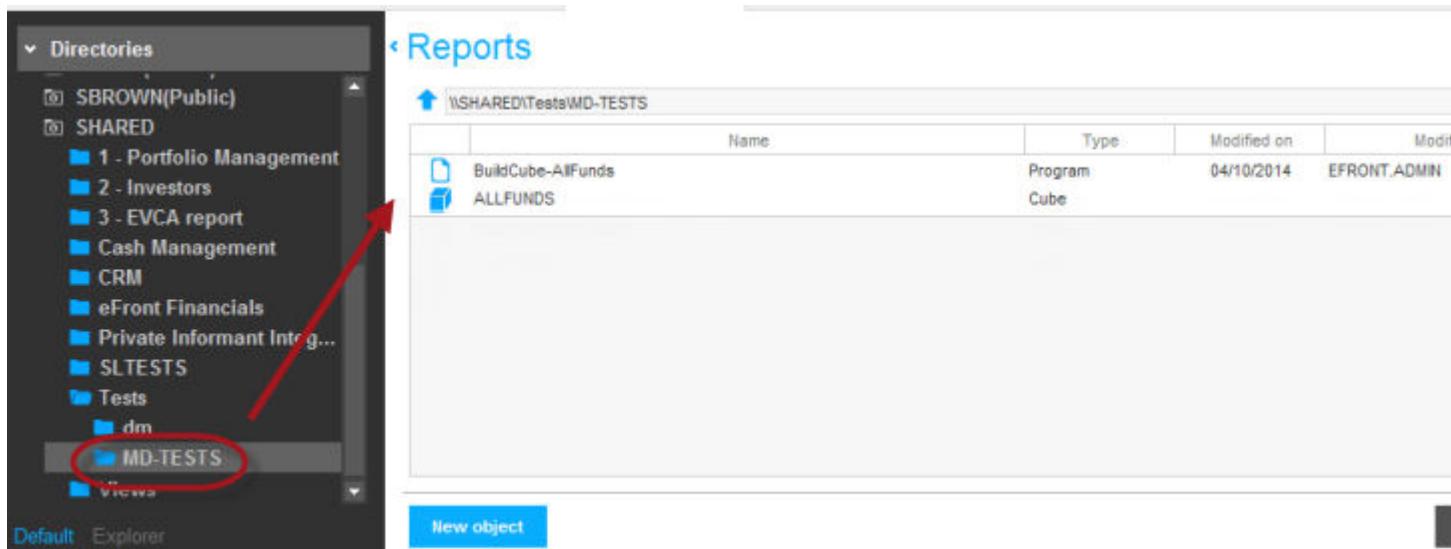
## 5.37.1.4. 1. Result - Program A

[BuildCube-AllFunds](#)

Execute									
	IQID	FUND	ABBREVIATION	FTYPE	FNATURE	FSTAGE	FREGION	STARTDATE	CLOSINODEATE
1	E024FB11816E4FFC	ABC_Basic	ABC_Basic	USA-LLC	20000000	PE	NA_USA		
2	15A7DFE086074AE	(CVC European	CVC EEP VI		20000000	PE_BO_LARGE	EUROPE		30/09/2012
3	ACAAAB5CDAD8344	eFront World	eFront WP PE						
4	2F9E42592F3D44DB	ANDROMEDA II	ANDROMEDA II		35000000	PE_BO	NA		31/01/2010
5	8D99FC5253DB4D2	(ANTLIA IV	ANTLIA IV		50000000	PE_MEZZ	AFRICA_SOUTHERN		30/04/2010
6	5C8EF008AADA4C2	ARA III	ARA III		45000000	PE_BO	EUROPE_WESTERN		31/12/2010
7	8CE7396F20774BD9	CAELUM I	CAELUM I		25000000	PE_VC	NA		28/02/2011
8	03FB893408A2477E	CARINA II	CARINA II		25000000	PE_BO	EUROPE_WESTERN		30/12/2011
9	74C72CB31C244D7	(CETUS VI	CETUS VI		40000000	PE_BO	EUROPE_NORTHERN		30/09/2011
10	58365BB6298E4F8C	DRACO I	DRACO I		50000000	PE_MEZZ	EUROPE_NORTHERN		30/11/2013
11	7AF5B70458DA476I	ANDROMEDA III	ANDROMEDA III		10000000	PE_BO	ASIA_WESTERN		31/01/2012
12	1F58C88EF2E14AFE	NORMA II	NORMA II		50000000	PE_BO	EUROPE_WESTERN		31/03/2012
13	B7B9EC1BB80B45D	ARA IV	ARA IV		10000000	PE_VC	NA		30/06/2013
14	06279586D507428F	LATINAO I	LATINAO I		15000000	PE_VC	LATINAMERICA		30/05/2013
15	0A8420151ABF48C	GRUS III	GRUS III		50000000	PE_BO	NA		28/09/2008
16	E78F744BAETF4FA	SCORPIUS	SCORPIUS		35000000	PE_BO	EUROPE_WESTERN		30/11/2008
17	21C3913F02BC46CE	ZETA	ZETA		45000000	PE_BO	AFRICA_CENTRAL		30/12/2008
18	F8B908FFF9EA43C	ΙΩΤΑ	ΙΩΤΑ		25000000	PE_VC	EUROPE_NORTHERN		07/02/2009
19	A4517F9E7A4346D	eFront IV Asia	Asia IV	GBR-LP		PE_GROWTH	ASIA	01/09/2009	04/10/2009
20	5114TECED5174E4CUK	Managers CI	UKM	GBR-GP					
21	58DA86571CA04EA	eFront Europe I	eFront Europe I		50000000	PE_VC	EUROPE_WESTERN		31/01/2010
22	A058A247E8A74C5	eFront America I	eFront America I		50000000	PE_VC	NA		30/04/2010
23	2B9EE483841E4ED4	eFront Asia I	eFront Asia I		50000000	PE_VC	ASIA_WESTERN		31/12/2010
24	45CSF8C832EC4D7	eFront World I	eFront World I		50000000	PE_VC	GLOBAL		28/02/2011
25	5C1579168CB74D84	eFront Europe II	eFront Europe II		50000000	PE_VC	EUROPE_WESTERN		31/01/2014
26	9EB378E12B83443B	Alpha Partners II	ALPHAII	USA-LP	40400000	RE	NA_USA	31/03/2005	05/04/2005
27	B21FBB57D290462F	Beta Fund L.P.	BETA	USA-LP	25000000	PE_SECONDARIES	NA_USA	17/06/2010	17/06/2010
28	4C419D97B0F0420E	Gamma Capital	GAMMAII	USA-LP	40000000	PE_SECONDARIES	GLOBAL	13/10/2006	13/11/2006
29	1E27C2E7AEE54A2	(Gamma Capital	GAMMAN	USA-LP	35200000	PE_VC	GLOBAL	10/12/2010	15/12/2010
30	AA61A422A5B947E	Delta Fund V, L.P.	DELTAV	USA-LP	25000000	PE_BO_MEDIUM	NA_USA	14/02/2007	14/02/2007
31	E36E18939D0B546A	Εpsilon Partners	EPSILONVII	USA-LP	20000000	PE_BO_LARGE	NA_USA	30/05/2007	30/05/2007
32	1770386D36B5492D	Zeta Partners IX	ZETAIIX	USA-LP		PE_BO_LARGE	NA_USA	10/01/2009	10/01/2009

The cube lists all the funds in the current database.

The program has created a global cube which is directly accessible from the directory of the current user:



and is stored on the eFront Cube server:

**SESSIONS**  
count = 2

Session Id	User Id	Time	Cubes count	Transactions count
D6B6074F-4AD2-4608-9D99-C9A34EF4C7E3	EFRONT.ADMIN	0	0	0
4CE6BBCA-41BC-4AF2-8832-305C61F26C17	EFRONT.ADMIN	0	0	0

**AVAILABLE GLOBAL CUBES ON DISK**  
count = 231

Cube Path	Columns Count	Rows Count	Physical Path
GLOBAL.CASHFLOWS	31	96095	\GLOBAL.CASHFLOWS
GLOBAL.SC_TR_INSTRU_HDG_DEBUG17072012	31	1610	\GLOBAL.SC_TR_INSTRU_HDG_DEBUG17072012
GLOBAL.SC_TR_PORTF_COMPARE_DEBUG17072012	27	176	\GLOBAL.SC_TR_PORTF_COMPARE_DEBUG17072012
GLOBAL.TREFM_POSITION_CHANGE_DEBUG17072012	18	29	\GLOBAL.TREFM_POSITION_CHANGE_DEBUG17072012
GLOBAL.CASHFLOWS	37	10000	\B5677FA3E23D47CBA025B5F90F5035DB\GLOBAL.CASHFLOWS
GLOBAL.CASHFLOWS	31	96095	\B5677FA3E23D47CBA025B5F90F5035DB\SUPERGIRL\GLOBAL.CASHFLOWS
GLOBAL.ALLFUNDS	14	92	\355651727DA9472686B1CB342210C688\GLOBAL.ALLFUNDS
GLOBAL.C_GLENTRIES_AS_FRONTCUBE_CUBE_GLOBAL	13	20	\355651727DA9472686B1CB342210C688\GLOBAL.C_GLENTRIES_AS_FRONT
GLOBAL.FUNDS	2	93	\355651727DA9472686B1CB342210C688\GLOBAL.FUNDS
GLOBAL.MC_DATASET_136	18	135	\355651727DA9472686B1CB342210C688\GLOBAL.MC_DATASET_136
GLOBAL.MC_TR_BMK_PORTF_HDG_EFRONT	23	7567	\355651727DA9472686B1CB342210C688\GLOBAL.MC_TR_BMK_PORTF_HD
GLOBAL.UNIVERSAL.CASHFLOWS	31	96095	\355651727DA9472686B1CB342210C688\GLOBAL.UNIVERSAL.CASHFLOWS
GLOBAL.VIRTUALPORT	4	18	\355651727DA9472686B1CB342210C688\GLOBAL.VIRTUALPORT
GLOBAL.ALLFUNDS	14	87	\355651727DA9472686B1CB342210C688\{SHARED}\Tests\MD-TESTS\GLOBA
GLOBAL.FUNDS	2	93	\355651727DA9472686B1CB342210C688\{SHARED}\test\GLOBAL.FUNDS
GLOBAL.FUNDS2	2	95	\355651727DA9472686B1CB342210C688\{SHARED}\test\GLOBAL.FUNDS2
GLOBAL.FUNDS3	2	95	\355651727DA9472686B1CB342210C688\{SHARED}\test\GLOBAL.FUNDS3

This cube is ready for usage in dynamic dashboards and/or eFront Excel Add-ins.

### 5.37.1.5. 2. Program B - Build the session cube that filters the global cube

LIBNAME USER ".;"

%LET GEO = "EUROPE\_WESTERN";

//build the session cube from global cube

DATA SESSION.ALLFUNDSFILTERED;

```
SET ALLFUNDS;  
/*Note: to increase performance of the session cube creation,  
you could also use the statement: EXTEND ALLFUNDS;*/  
RUN;  
//Filter the session cube  
PROC SETFILTER DATA=SESSION.ALLFUNDSFILTERED  
FILTER=(FREGION=%GEO);  
RUN;  
//For illustration purpose, print the cube  
PROC PRINT DATA=SESSION.ALLFUNDSFILTERED;  
RUN;
```

### 5.37.1.6. 2. Result: Program B

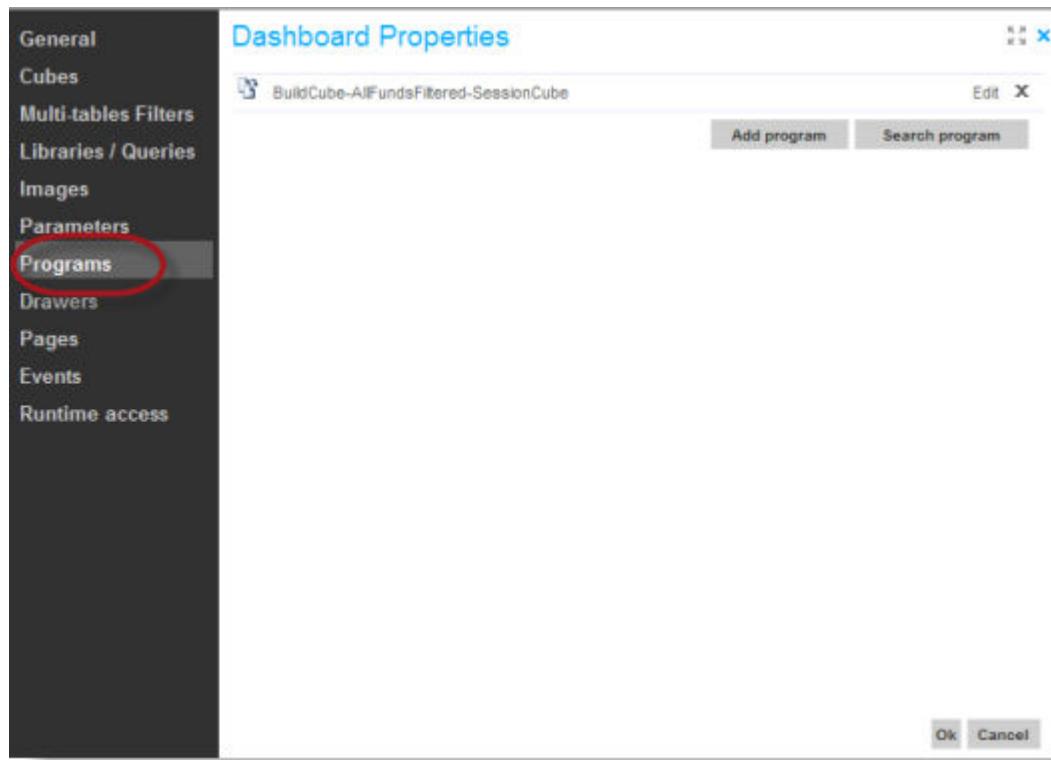
When we compile the program B from within the eFront Cube program editor, we still have the same cube as the one produced with program A:

## ↳ BuildCube-AllFundsFiltered-SessionCube

Execute								
	IQID	FUND	ABBREVIATION	FTYPE	FNATURE	FSTAGE	FREGION	STARTDATE
1	E024FB11016E4FFC	ABC_Basic	ABC_Basic	USA-LLC	20000000	PE	NA_USA	
2	15A7DFE006074AE	CVC European	CVC EEP VI		20000000	PE_BO_LARGE	EUROPE	
3	ACAAAB5CDAD8344	eFront World	eFront WP PE					
4	2F9E42592F3D44DB	ANDROMEDA II	ANDROMEDA II		35000000	PE_BO	NA	
5	8D99FC5253DB4D2I	ANTLIA IV	ANTLIA IV		50000000	PE_MEZZ	AFRICA_SOUTHERN	
6	5C6EF008AADA4C2ARA	II	ARA II		45000000	PE_BO	EUROPE_WESTERN	
7	8CE7396F20774BD9	CAELUM I	CAELUM I		25000000	PE_VC	NA	
8	03FB693408A2477E	CARINA II	CARINA II		25000000	PE_BO	EUROPE_WESTERN	
9	74C72CB31C244D7	CETUS VI	CETUS VI		40000000	PE_BO	EUROPE_NORTHERN	
10	58365BB6298E4F8CDRACO	I	DRACO I		50000000	PE_MEZZ	EUROPE_NORTHERN	
11	7AF5B704580A478I	ANDROMEDA II	ANDROMEDA II		10000000	PE_BO	ASIA_WESTERN	
12	1F58C88EF2E14AFE	NORMA II	NORMA II		50000000	PE_BO	EUROPE_WESTERN	
13	B7B9EC1B8B0B45D	ARA IV	ARA IV		10000000	PE_VC	NA	
14	06279586D507428F	LATINAO I	LATINAO I		15000000	PE_VC	LATINAMERICA	
15	0A8420151ABF48C	GRUS II	GRUS II		50000000	PE_BO	NA	
16	E78F744BAE7F4FA	SCORPIUS	SCORPIUS		35000000	PE_BO	EUROPE_WESTERN	
17	21C3913F02BC46CE	ZETA	ZETA		45000000	PE_BO	AFRICA_CENTRAL	
18	F8B908FFF9EA43C	IOTA	IOTA		25000000	PE_VC	EUROPE_NORTHERN	
19	A4517F9E7A4346D	eFront IV Asia	Asia IV	GBR-LP		PE_GROWTH	ASIA	01/09/2009
20	51147ECED5174E4CUK	Managers CI	UKM	GBR-GP				
21	5B0A86571CA04EA	eFront Europe I	eFront Europe I		50000000	PE_VC	EUROPE_WESTERN	
22	A058A247E8A74C5	eFront America I	eFront America I		50000000	PE_VC	NA	
23	2B9EE4B3841E4ED4	eFront Asia I	eFront Asia I		50000000	PE_VC	ASIA_WESTERN	
24	45C5F8C832EC4D7	eFront World I	eFront World I		50000000	PE_VC	GLOBAL	
25	5C1579168CB74D84e	eFront Europe III	eFront Europe III		50000000	PE_VC	EUROPE_WESTERN	
26	9EB378E12B83443B	Alpha Partners II	ALPHAII	USA-LP	40400000	RE	NA_USA	31/03/2005
27	504488E7000010000000	BETA	BETA	USA-LO	25000000	PE_SECONDARIES	NA_USA	17/02/2010

### 5.37.1.7. 3. Link cubes and program to dashboard

We link the eFront Cube program that generates the session cube to the dashboard.

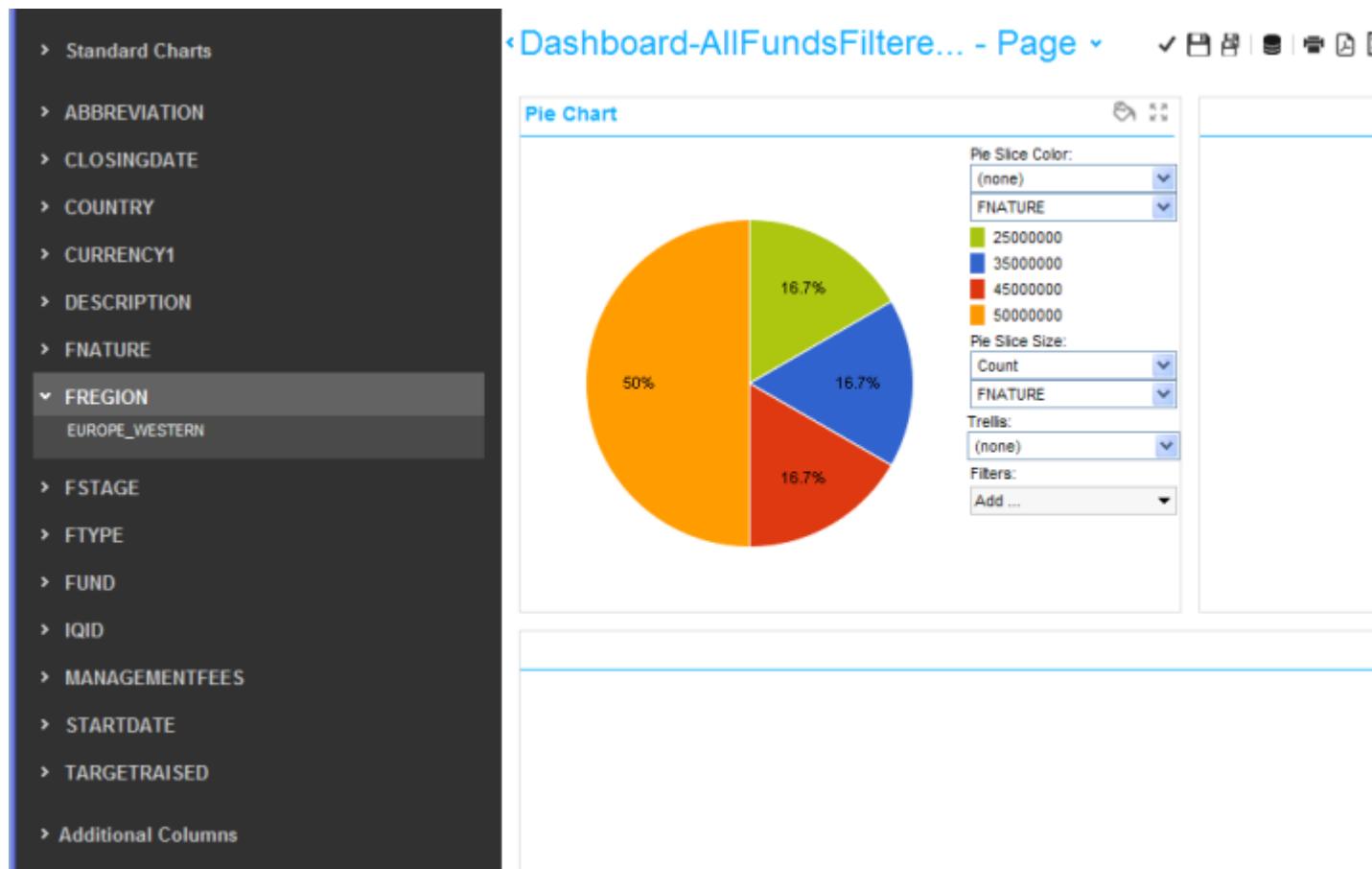


We also link the **session cube** to the dashboard.

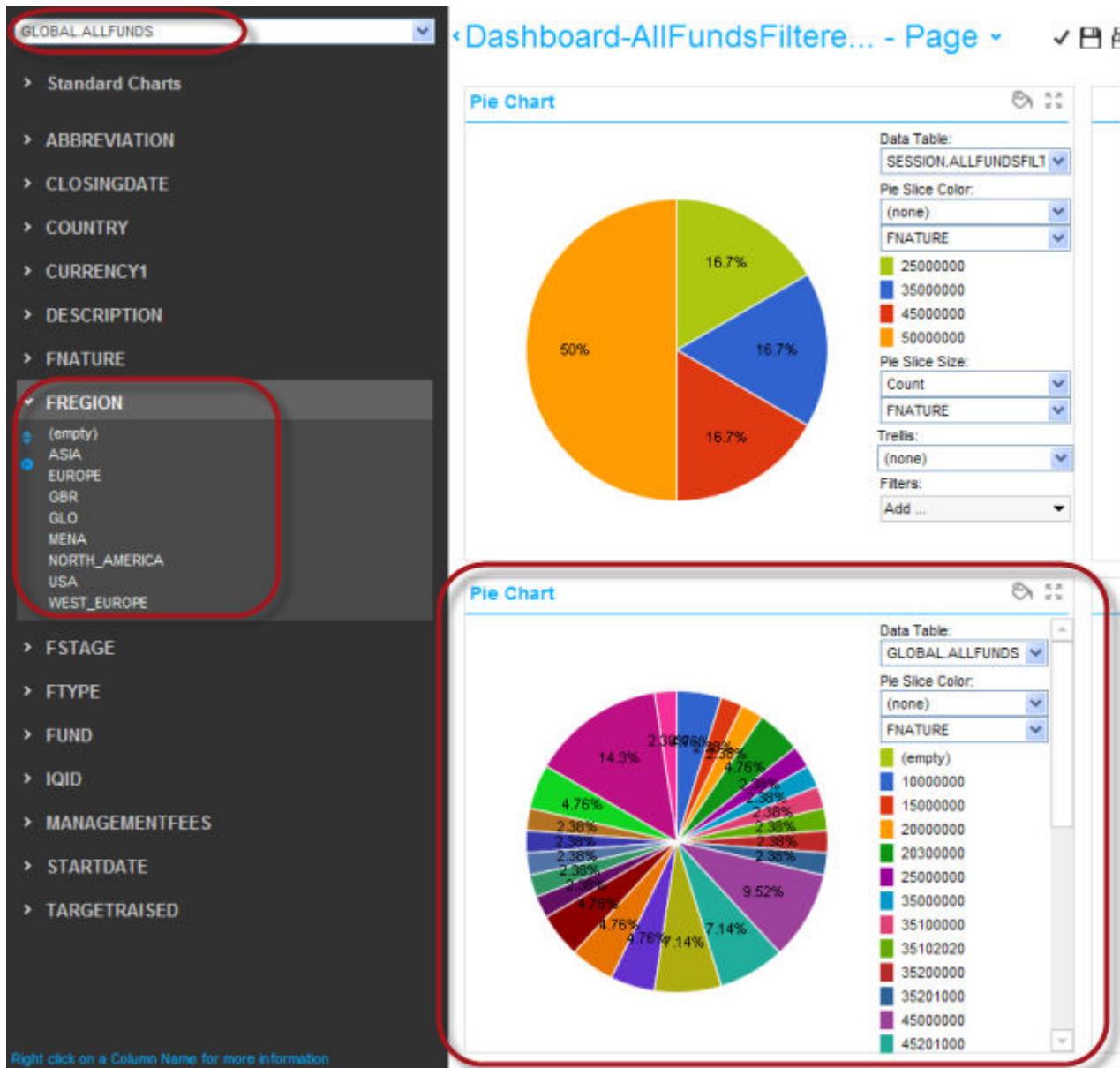


**i** The cube is only accessible from within the dashboard for selection, if you have executed the eFront Cube program just before, which ensures that the session cube has been stored in the memory.

And now, if we refresh the dashboard, we get only the funds that correspond to the filtering option:



For illustration purpose only: If we would link the unfiltered global cube directly to the dashboard, we would get all the funds:



## 5.38. Examples - XML Schema Definition

The following examples illustrate how to use the eFront specific nodes that are evaluated by the parser:

- [EF\\_LOOP-EF\\_VALUE-EF\\_ATTRIB-EF\\_G-EF\\_MACRO](#)
- [EF\\_SELECT-EF\\_CASE-EF\\_OTHERWISE](#)

### 5.38.1. EF\_LOOP-EF\_VALUE-EF\_ATTRIB-EF\_G-EF\_MACRO

#### 5.38.1.1. Goal

Build an XML template that allows to generate an XML report that lists a list of funds, and for each fund, displays the portfolio companies.

#### 5.38.1.2. Features being illustrated

- [eFront Invest XML Schema Definition](#)
- [EF\\_LOOP - EF\\_VALUE](#)
- [EF\\_ATTRIB](#)
- [EF\\_G](#)
- [EF\\_MACRO](#)

#### 5.38.1.3. XML Template

```
<FundList>

<ef_loop table = "Funds">

<Fund>

<ef_attrib name = "name"><ef_value column = "Fund_Name"/></ef_attrib>

<ef_loop table = "Companies" where="Investor_ID=?Fund_ID">

<PortfolioCompanyName><ef_value column = "Company_Name"/></
PortfolioCompanyName>

</ef_loop>
```

```
<FundSize>  
<ef_value column = "Fund_Size" format = "#,###.00"/>  
</FundSize>  
  
<ValuationDate>  
<ef_g column = "Reference_Date" table = "Transactions" where ="Transaction_Type =  
'VAL'; Security_ID = ?Security_ID" format = "dd-mm-yyyy"/>  
</ValuationDate>  
  
<ClosingDate>  
<ef_macro name = "Closing_Date" format = "dd-mm-yyyy"/>  
</Closingdate>  
  
</Fund>  
  
</ef_loop>  
</FundList>
```

#### 5.38.1.4. XML File being built with the Template

```
<FundList>  
  
<Fund name = "Tamino Fund III">  
  
<PortfolioCompanyName>Company A</PortfolioCompanyName>  
  
<PortfolioCompanyName>Company A</PortfolioCompanyName>  
  
<FundSize>123,444,300.00</FundSize>  
  
</Fund>  
  
<Fund name = "Tamino Fund IV">  
  
<PortfolioCompanyName>Company D</PortfolioCompanyName>
```

```
<FundSize>321,300,444.00</FundSize>
<ValuationDate>31-12-2008</ValuationDate>
<ClosingDate>31-12-2008</ClosingDate>
</Fund>
</ef_loop>
</FundList>
```

## 5.38.2. EF\_SELECT-EF\_CASE-EF OTHERWISE

### 5.38.2.1. Goal

Build an XML template that allows to generate an XML report that lists information about instruments according to the instrument type.

### 5.38.2.2. Features being illustrated

- eFront Invest XML Schema Definition
- EF\_SELECT - EF\_CASE - EF OTHERWISE
- EF\_ATTRIB

### 5.38.2.3. XML Template

```
<Instrument>
<ef_select expr = "Security_Type_Code">
<ef_case value = "1">
<ef_attrib name = "Type">Loan</ef_attrib>
</ef_case>
<ef_case value = "2">
<ef_attrib name = "Type">Shares</ef_attrib>
<NBShares><ef_value column = "NB_Shares" format = "#,###"/></NBShares>
```

```
</ef_case>

EF_OTHERWISE

<ef_attrib name = "Type">Other</ef_attrib>

</ef_otherwise>

</ef_select>

</Instrument>
```

#### 5.38.2.4. XML File being built with the Template

The template above evaluates into any of the following:

```
<Instrument Type = "Loan">

</Instrument>
```

Or

```
<Instrument Type = "Shares">

<NBShares>34000</NBShares>

</Instrument>

</Instrument>
```

Or

```
<Instrument Type = "Other">

</Instrument>

<Fund name = "Tamino Fund III">

<PortfolioCompanyName>Company A</PortfolioCompanyName>

<PortfolioCompanyName>Company A</PortfolioCompanyName>
```

```
<FundSize>123,444,300.00</FundSize>
</Fund>

<Fund name = "Tamino Fund IV">
<PortfolioCompanyName>Company D</PortfolioCompanyName>
<FundSize>321,300,444.00</FundSize>
<ValuationDate>31-12-2008</ValuationDate>
<ClosingDate>31-12-2008</ClosingDate>
</Fund>
</ef_loop>
</FundList>
```

## 6. eFront Script - Calling the Calculation Engine

### Concept

## 6.1. About eFront Invest calculation engines

## 6.2. Accessing CE tables from within an eFront Script program

In the following we explain the principles of accessing the **eFront Invest** calculation engines from within an **eFront Script** program. There is no difference between an **eFront Cube - eFront Script program**, and an **eFront Report - eFront Script program** calling the **eFront Invest** calculation engine, as far as the syntax and the underlying working principle is concerned.

- [Integrating CE tables into eFront Script programs](#)
- [Macros and variables to be used when calling CE tables](#)

### 6.3. Data table description

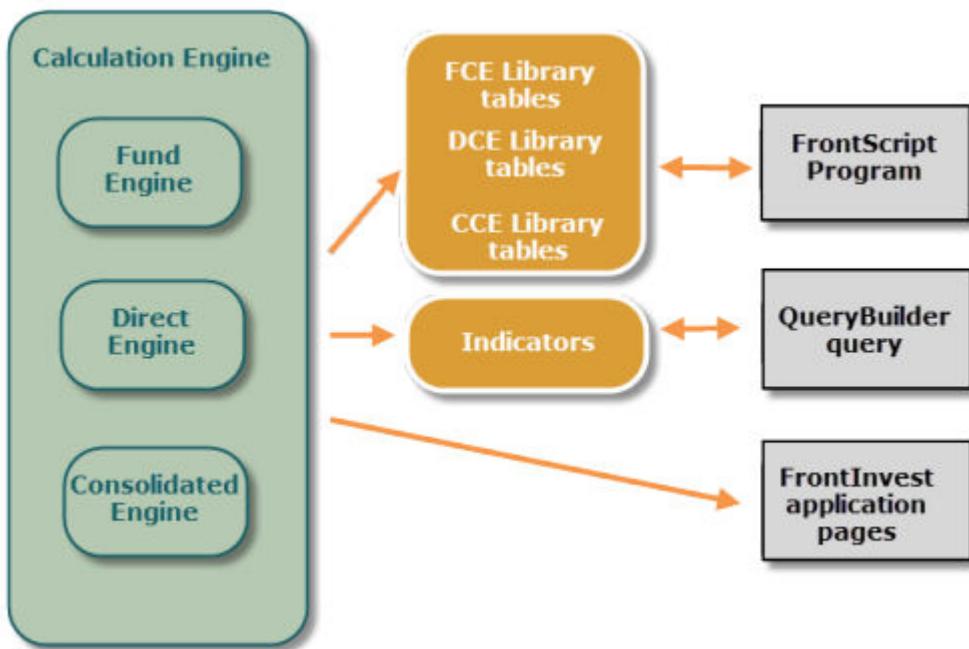
- Data tables produced by Direct Calculation Engine
- Data tables produced by the Funds Calculation Engine
- Data tables produced by the Consolidated Calculation Engine

## 6.4. Examples

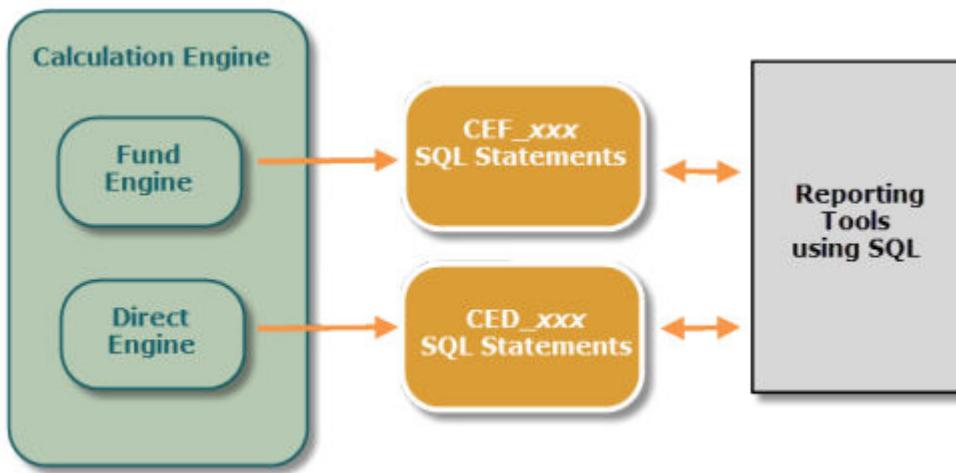
### Examples - Calculation Engine Tables

## 6.5. About eFront Invest calculation engines

Up to **eFront Invest V9.0**, the eFront Invest calculation engine is written in **VB.DotNet**. Its constituents (Direct Calculation Engine, Fund Calculation Engine, Consolidated Engine) are called through eFront Script calls (using the FCE, DCE, and CCE libraries), QueryBuilder queries (indicators), or directly from eFront Invest.



Up to **eFront Invest V9.0**, the eFront Invest calculation engine is written in **SQL**. Therefore, the constituents of the SQL Engine (Direct Calculation Engine, Fund Calculation Engine) can be called from any SQL based reporting tool such as **SSRS** and **Crystal Reports**, but also **eFront Analytics**, **eFront Excel** and even **eFront Report**.



Two major benefits underly the migration from a VB.DotNet written engine to an SQL based engine:

- Make the data accessible from a large variety of reporting tools
- Increase the capacity of managing huge data volumes

The calculation engine can be located on the database server or a dedicated server.

**i** In **eFront Invest V9.0**, the calculations done by the product use still the VB.DotNet written engine, but the SQL engine is available for third party tools!

Up **from eFront Invest V9.1**, the VB.DotNet written engine will be entirely replaced by the SQL written engine, or users might be able to switch from one engine to the other. The switch will be transparent for users, meaning the eFront Script procedures will access the new calculation engine without any syntactical changes to be made.

## 6.6. Integrating CE tables into eFront Script programs

All the tables produced by the calculation engines are accessible like regular **eFront Report** tables through specific library aliases:

- DCE for direct calculation engine
- FCE for Fund calculation engine
- CCE for Consolidated calculation engine

### 6.6.1. Available tables - DCE

- **DIRECTCASHFLOWS**: cash flows linked to direct investments
- **INSTRUMENTPOSITIONS**: positions of instruments
- **REFINSTRUMENTPOSITIONS**: aggregates results per reference instrument; returns the positions of all the investment instruments related to the same reference instrument
- **INVESTMENTPOSITIONS**: positions of investments
- **INVESTORPOSITIONS**: positions by investor
- **INSTRUMENTROUNDS**: investments for the instrument can be done in successive rounds; the table returns instrument positions per investment round
- **COMPANYPOSITIONS**: positions by company
- The calculation engine generates the following tables:
  - FV\_T\_PORTCO\_TRANSACTIONS
  - FV\_T\_PORTCO\_POSITIONS\_BY\_SECURITIES
  - FV\_T\_PORTCO\_POSITIONS\_BY\_INVESTMENT
  - FV\_T\_PORTCO\_POSITIONS\_BY\_INVESTORS

### 6.6.2. Available tables - FCE

- **FUNDOPERATIONS, FUNDINVESTOROPERATIONS**: fund operations and fund operations by investors
- **FUNDSHAREOPERATIONS, FUNDINVESTORSHAREOPERATIONS**: fund operations by share and fund operations by share and by investor
- **FUNDPOSITIONS, FUNDPOSITIONSWITHOUTCARRIED**: fund positions with and without carried

- **FUNDINVESTORPOSITIONS:** positions of fund investors
- **FUNDSHAREPOSITIONS, FUNDINVESTORSHAREPOSITIONS:** fund's shares positions and fund's shares positions by investors
- **FUNDCASHFLOWS:** cash flows linked to fund investments
- **INVESTORPOSITIONS:** positions by investor
- **MASTERFUNDINVESTORPOSITIONS:** only to be used if the account manages master funds; returns the global position of the master fund.
- The calculation engine generates the following tables:
  - FV\_T\_FUND\_TRANSACTIONS
  - FV\_T\_FUND\_SEC\_POSITIONS\_BY\_INVESTMENTS
  - FV\_T\_FUND\_SEC\_POSITIONS\_BY\_FUNDS
  - FV\_T\_FUND\_POSITIONS\_BY\_FUNDS
  - FV\_T\_FUND\_POSITIONS\_BY\_INVESTMENTS
  - FV\_T\_FUND\_POSITIONS\_BY\_INVESTORS
  - FV\_T\_FUND\_OPERATIONS\_BY\_INVESTORS
  - FV\_T\_FUND\_OPERATIONS\_BY\_FUNDS

### 6.6.3. Available tables - CCE

- **INVESTMENTPOSITIONS:** consolidated view of investment positions
- **INVESTORPOSITIONS:** consolidated view of investor positions

### 6.6.4. Integration of CE tables into eFront Script program

You can call **DCE**, **FCE** and **CCE** tables like any other regular **eFront Report** library tables. DCE, FCE and CCE are reserved **eFront Report** library **aliases**.

### 6.6.5. Examples

#### Examples - Calculation Engine Tables



- CE tables are read-only, they cannot be used as output tables
- All options like DROP, KEEP, RENAME and WHERE work properly.

## 6.7. Macros and variables to be used when calling CE tables

You can modify the default behavior of the calculation engines applied when building the CE tables. To do so, you can use different macros. In the following you find a summary table of all the available macros. For details about each macro, refer to the following topics:

[Macros and variables to be used when calling DCE tables](#)

[Macros and variables to be used when calling FCE tables](#)

[Macros and variables to be used when calling CCE tables](#)

### 6.7.1. Examples of macro usage

[Examples - Calculation Engine Tables](#)

Macro	DCE	FCE	CCE
DATE	X	X	X
STARTDATE	X	X	X
USE_DRAFT	X	X	X
USE_REAL_DUE_INTEREST	X		
USE_INVESTEE_CURR	X		X
USE_INVESTOR_CURR		X	
USE_SHARE_CURR		X	
CE_CURRENCY	X		
CALCULATE_IRR	X	X	
CALCULATE_TWR	X	X	
ANNUALIZED_TWR	X	X	
CALCULATE SHARES	X		
CALCULATE_ACCRUALS	X		
USE_EQUALIZATION		X	
EQUALIZEDDATE		X	
USE_FUNDPOSITIONS		X	
USE_FUNDPOSITIONSWITHOUTCARRIED		X	
USE_FUNDOPERATIONS		X	

USE_FUNDSHAREPOSITIONS		X	
USE_FUNDSHAREOPERATIONS		X	
USE_FUNDINVESTORPOSITIONS		X	
USE_FUNDINVESTOROPERATIONS		X	
USE_FUNDINVESTORSHAREPOSITIONS		X	
USE_FUNDINVESTORSHAREOPERATIONS		X	
IQID_FUND	X		X
IQID_COMPANY	X		X
IQID_INVESTOR	X		X
PORTFOLIO_REPORT	X		
A_COMPANY	X		
FOF_INVESTOR_REPORT		X	
FOF_INVESTOR_PORTFOLIO_REPORT		X	
IQID_FOF_FUND		X	
IQID_MASTER_FUND		X	
IQID_FOF_INVESTOR		X	
FOF_GL_CHART_REPORT		X	
IQID_CHART		X	
VALUATION_CALCULATION_DATES	X		
BY_TRANSPARENCY	X	X	

## 6.7.2. Macros and variables to be used when calling DCE tables

### 6.7.2.1. Applicable macros

You can modify the default behavior of the calculations done by the Direct Calculation Engine when calling the DCE tables from within eFront Script steps by using the following macros:

- If the DATE macro is not defined, the default is Today.
- If you define the STARTDATE macro, the system will take the value defined for the start date, and perform calculations for all the transactions up from this date.
- If you define the USE\_DRAFT macro, draft transactions are used by the calculation engine. Otherwise, they are ignored.
- If you define a value for the variable CE\_CURRENCY macro, this value will be used as currency (e.g. 'EUR', 'USD'). Otherwise, calculations are done in investor currency.

- If you define the USE\_INVESTEE\_CURR macro, calculations are done in investee currency. Otherwise, calculations are done in investor currency.



By default, the direct calculation engine, when it is called from **eFront Script** (by using DCE.tables), does not calculate ND/FD ownership, PIK/cash accruals and IRR. This behavior has been set this way to ensure highest performance of the DCE, to decrease memory used, and calculation time. Therefore, if you need ND/FD ownership, PIK/cash accruals and IRR, you need to use any of the macros below:

- If you define the CALCULATE SHARES macro, calculations include ND and FD ownership. Otherwise, the system does not calculate ND and FD ownership. Please note that this macro can also work in FrontCube mode.
- If you define the CALCULATE\_ACCRUALS macro, calculations include PIK and cash accruals. Otherwise, the system does not calculate accruals.
- If you define the CALCULATE\_IRR macro, calculations include the IRR. Otherwise, the system does not calculate the IRR. Please note that this macro can also work in FrontCube mode.
- If you define the CALCULATE\_TWR macro, calculations include the TWR. Otherwise, the system does not calculate the TWR.
- If you define the ANNUALIZED\_TWR macro, calculations include the Annualized TWR. Otherwise, the system does not calculate the Annualized TWR.



By default, the direct calculation engine calculates data for all the investors and all the investees. You can modify this behavior and filter the list of investors and investees using the macros below:

- If you define the PORTFOLIO\_REPORT macro, calculations are done on behalf of the investor(s) whose IQID(s) is(are) transmitted.  
Set at least one of the following variables:
  - Set the variable IQID\_COMPANY to the investor companies IQID (collection or unique value).
  - Set the variable IQID\_FUND to the investor funds IQID (collection or unique value).
  - Set the variable IQID\_INVESTOR to the investors IQID (collection or unique value).
- If you define the A\_COMPANY macro, calculations are done on behalf of the investee(s) whose IQID(s) is(are) transmitted.

- Set the variable IQID\_COMPANY to the invested companies IQID (collection or unique value).
- If you define the VALUATION\_CALCULATION\_DATES macro, when calling the calculation engine and getting the DIRECTCASHFLOWS table, the system adds a row on every day, on each investment instrument (instruments of all types) in order to adjust the valuation. Note that the system does not add any dummy transactions before the first transaction occurring on the investment instrument.

### 6.7.2.2. Examples of macro usage

Examples - Calculation Engine Tables

### 6.7.3. Macros and variables to be used when calling FCE tables

#### 6.7.3.1. Applicable macros

You can modify the default behavior of the calculations done by the **Fund Calculation Engine** when calling the FCE tables from within **eFront Script** steps by using the following macros:

- If the DATE macro is not defined, the default is Today.
- If you define the STARTDATE macro, the system will take the value defined for the start date, and perform calculations for all the fund operations up from this date.
- If you define the USE\_DRAFT macro, fund draft operations are used by the calculation engine. Otherwise, they are ignored.
- If you define the USE\_INVESTOR\_CURR macro, calculations are done in investor currency. Otherwise, calculations are done in fund currency.
- If you define the USE\_SHARE\_CURR macro, calculations are done in share currency, otherwise calculations are done in fund currency.



#### NOTE

If both USE\_INVESTOR\_CURR and USE\_SHARE\_CURR macros are used, the USE\_INVESTOR\_CURR will be ignored.

- If you define the IQID\_MASTER\_FUND macro, the system returns the global position of the master fund(s), whose ID is passed along to the FCE engine. This obviously only applies if you handle master funds.

- Set the variable IQID\_MASTER\_FUND to the investee fund IQID (unique value).
- If you define the USE\_EQUALIZATION macro, the system returns equalized positions per fund investor.
- If you define the EQUALIZEDDATE macro, the system returns the date at which you have the view of equalizations.
- If you define the BY\_TRANSPARENCY macro, the system gets the FCE positions by transparency. This macro can work with or without the FrontCube option.



By default the **Fund Calculation Engine**, when it is called from **eFront Script** (by using FCE. tables), extracts all the data from the database, creates data structures and then inquires these data structures to build the FCE tables. This default behavior takes up a lot of memory space and calculation time. Therefore, you can restrict the data to be extracted right from scratch, and tell the engine what type of data you need. To do so, you can use any of the macros below. Performance will be greatly improved.

- If you define the USE\_FUNDPOSITIONS macro, data are populated for the FUNDOPERATIONS and FUNDPOSITIONS tables.
- If you define the USE\_FUNDPOSITIONSWITHOUTCARRIED, data are populated for the FUNDOPERATIONS and FUNDPOSITIONSWITHOUTCARRIED tables.
- If you define the USE\_FUNDOPERATIONS, data are populated for the FUNDOPERATIONS table.
- If you define the USE\_FUNDSHAREPOSITIONS, data are populated for the FUNDSHAREPOSITIONS table.
- If you define the USE\_FUNDSHAREOPERATIONS, data are populated for the FUNDOPERATIONS, FUNDSHAREOPERATIONS and FUNDSHAREPOSITIONS tables.
- If you define the USE\_FUNDINVESTORPOSITIONS, data are populated for the FUNDINVESTORPOSITIONS table.
- If you define the USE\_FUNDINVESTOROPERATIONS, data are populated for the FUNDSHAREOPERATIONS and FUNDINVESTOROPERATIONS tables.
- If you define the USE\_FUNDINVESTORSHAREPOSITIONS, data are populated for the FUNDINVESTORSHAREPOSITIONS table.
- If you define the USE\_FUNDINVESTORSHAREOPERATIONS, data are populated for the FUNDINVESTORSHAREOPERATIONS and FUNDSHAREPOSITIONS tables.
- If you define the CALCULATE\_IRR macro, calculations include the IRR. Otherwise, the system does not calculate the IRR.

- If you define the CALCULATE\_TWR macro, calculations include the TWR. Otherwise, the system does not calculate the TWR.

 By default, the fund calculation engine calculates positions for all the investee funds in the database. You can modify this behavior and filter the list of investee funds using the macros below:

- If you define the FOF\_INVESTOR\_REPORT macro, calculations are done on behalf of the fund(s) whose IQID(s) is(are) transmitted, and the system returns the data regarding the relationship between the fund and its investors.  
Set at least one of the following variables:
  - Set the variable %IQID\_FOF\_FUND to the invested fund IQID (collection or unique value).
  - Set the variable %IQID\_MASTER\_INVESTOR to the invested master fund IQID (collection or unique value).
- If you define the FOF\_INVESTOR\_PORTFOLIO\_REPORT macro, calculations are done on behalf of the investor(s) whose IQID(s) is(are) transmitted, and the system returns the data regarding the relationship between the investor and its invested funds.
  - Set the variable %IQID\_FOF\_INVESTOR to the investor IQID (collection or unique value).
- If you define the FOF\_INVESTOR\_REPORT macro, calculations are done on behalf of the fund(s) whose IQID(s) is(are) transmitted, and the system returns the data regarding the relationship between the fund and its investors.
  - Set the variable IQID\_FOF\_FUND to the invested fund IQID (collection or unique value).
- Provided the previous macro has not been defined, if you define the FOF\_INVESTOR\_PORTFOLIO\_REPORT macro, calculations are done on behalf of the investor(s) whose IQID(s) is(are) transmitted, and the system returns the data regarding the relationship between the investor and its invested funds.
  - Set the variable IQID\_FOF\_INVESTOR to the investor IQID (collection or unique value).
- In addition to the aforesaid macros, if you define the FOF\_GL\_CHART\_REPORT macro and the IQID\_CHART variable, you can reduce the scope of the fund operations used to the ones included in the chart.

- Set the variable IQID\_CHART to the GL Chart IQID (unique value).

### 6.7.3.2. Examples of macro usage

Examples - Calculation Engine Tables

## 6.7.4. Macros and variables to be used when calling CCE tables

### 6.7.4.1. Applicable macros

You can modify the default behavior of the calculations done by the **Consolidated Calculation Engine** when calling the CCE tables from within **eFront Script** steps by using the macros below.

- If the DATE macro is not defined, the default is Today.
- If you define the STARTDATE macro, the system will take the value defined for the start date, and perform calculations for all the fund operations and transactions up from this date.
- If you define the USE\_DRAFT macro, fund draft operations and investment draft transactions are used by the calculation engine. Otherwise, they are ignored.
- If you define the USE\_INVESTEE\_CURR macro, calculations on the consolidated portfolio are done in investee currency. Otherwise, calculations are done in fund currency.



By default, data are calculated for the investor whose IQID is passed along to the CCE via the IQID\_FUND, IQID\_COMPANY or IQID\_INVESTOR.

- Set the variable IQID\_COMPANY to the investor companies IQID (collection or unique value).
- Set the variable IQID\_FUND to the investor funds IQID (collection or unique value).
- Set the variable IQID\_INVESTOR to the investors IQID (collection or unique value).

### 6.7.4.2. Examples of macro usage

Examples - Calculation Engine Tables

## 6.8. Data tables produced by Direct CE (DCE)

The **eFront Invest** calculation engine for direct investments produces the following tables:

- [DirectCashflows](#)
- [Section 6.8.2, “InstrumentPositions”](#)
- [Section 6.8.3, “RefInstrumentPositions”](#)
- [Section 6.8.4, “InvestmentPositions”](#)
- [InvestorPositions](#)
- [InstrumentRounds](#)
- [CompanyPositions](#)

The calculation engine generates the following tables:

- [FV\\_T\\_PORTCO\\_TRANSACTIONS](#)
- [FV\\_T\\_PORTCO\\_POSITIONS\\_BY\\_SECURITIES](#)
- [FV\\_T\\_PORTCO\\_POSITIONS\\_BY\\_INVESTMENT](#)
- [Section 6.8.11, “FV\\_T\\_PORTCO\\_POSITIONS\\_BY\\_INVESTORS”](#)

### 6.8.1. DirectCashFlows

 When you call **DIRECTCASHFLOWS** from the **CalcEngine**, it generates by default a virtual transaction of adjustment at the report date. You can use this virtual transaction for example to calculate the sum of **UNREALIZEDGAINLOSS** at the report date.

#### 6.8.1.1. DIRECTCASHFLOWS

```
New CECubeColumn() { _
```

```
    New CECubeColumn("TRANSACTION_ID", "ID transaction{F}ID transaction",
        GetType(String)), _
```

```
    New CECubeColumn("CURRENCY", "Currency{F}Devise", GetType(String)), _
```

```
    New CECubeColumn("REFERENCE_DATE", "Reference date{F}Date de référence",
        GetType(DateTime)), _
```

New CECubeColumn("AMOUNT", "Amount", GetType(Double)), \_

New CECubeColumn("ACCOUNTING\_DATE", "Accounting Date", GetType(DateTime)), \_

—

New CECubeColumn("IRR\_DATE", "IRR date{F}Date TRI", GetType(DateTime)), \_

New CECubeColumn("DRAFT", "Draft{F}Brouillon", GetType(Boolean)), \_

New CECubeColumn("TRANSACTION\_DC", "D/C{F}D/C", GetType(String)), \_

New CECubeColumn("DESCRIPTION", "Description", GetType(String)), \_

New CECubeColumn("TRANSACTION\_TYPE", "Transaction type{F}Type transaction", GetType(String)), \_

New CECubeColumn("TRANSACTION\_INCOMMITMENT", "Transaction in commitment{F}Transaction dans engagement", GetType(Boolean)), \_

New CECubeColumn("TRANSACTION\_ISVALOOP", "Valuation Transaction{F}Transaction valorisation", GetType(Boolean)), \_

New CECubeColumn("SECURITY\_ID", "ID Investment Security{F}ID titre investissement", GetType(String)), \_

New CECubeColumn("SECURITY\_NAME", "Security name{F}Nom du titre", GetType(String)), \_

New CECubeColumn("SECURITY\_CURRENCY", "Security currency{F}Devise du titre", GetType(String)), \_

New CECubeColumn("SECURITY\_STATUS", "Security status{F}Statut du titre", GetType(String)), \_

New CECubeColumn("SECURITY\_COUNTRY", "Security country{F}Pays du titre", GetType(String)), \_

New CECubeColumn("SECURITY\_CLASS\_NAME", "Security class{F}Classe de titre", GetType(String)), \_

```
New CECubeColumn("SECURITY_TYPE_CODE", "Type", GetType(String)), _  
New CECubeColumn("SECURITY_GROUPBY", "InstrumentGROUPBY",  
GetType(String)), _  
New CECubeColumn("SECURITY_ISIN", "Security ISIN code{F}Code ISIN du titre",  
GetType(String)), _  
New CECubeColumn("SECURITY_TICKER", "Security Ticker code{F}Code Ticker du  
titre", GetType(String)), _  
New CECubeColumn("INSTRUMENTFATHER_ID", "InstrumentFATHERID",  
GetType(String)), _  
New CECubeColumn("FATHERINSTRSTARTDATE", "fatherInstrSTARTDATE",  
GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRTERMDATE", "fatherInstrTERMDATE",  
GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRSTARTREPDATE", "fatherInstrSTARTREPDATE",  
GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRCLOSEREPDATE",  
"fatherInstrCLOSEREPDATE", GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRMARKET", "fatherInstrMARKET",  
GetType(String)), _  
New CECubeColumn("FATHERINSTRBASERATE", "fatherInstrBASERATE",  
GetType(Double)), _  
New CECubeColumn("FATHERINSTRINTERESTRATE", "fatherInstrINTERESTRATE",  
GetType(Double)), _  
New CECubeColumn("FATHERINSTRCONVERSIONRATE",  
"fatherInstrCONVERSIONRATE", GetType(Double)), _  
New CECubeColumn("FATHERINSTRSTARTINTERESTDATE",  
"fatherInstrSTARTINTERESTDATE", GetType(DateTime)), _
```

```
New CECubeColumn("FATHERINSTRCLOSINGINTERESTDATE",
"fatherInstrCLOSINGINTERESTDATE", GetType(DateTime)), _

New CECubeColumn("PROPERTY_ID", "Property ID", GetType(String)), _

New CECubeColumn("COMPANY_ID", "ID Company{F}ID société", GetType(String)), _

New CECubeColumn("COMPANY_CURR", "Company currency{F}Devise société",
GetType(String)), _

New CECubeColumn("COMPANY_NAME", "Company name{F}Nom société",
GetType(String)), _

New CECubeColumn("COMPANY_COUNTRY", "Company country{F}Pays société",
GetType(String)), _

New CECubeColumn("COMPANY_INVESTTYPE", "Company investment type{F}Type
investissement société", GetType(String)), _

New CECubeColumn("COMPANY_STAGE", "Company stage{F}Devise société",
GetType(String)), _

New CECubeColumn("COMPANY_INSTRUMENTNAME",
"InvesteeINSTRUMENTNAME", GetType(String)), _

New CECubeColumn("INVESTOR_ID", "ID investor{F}ID investisseur",
GetType(String)), _

New CECubeColumn("INVESTOR_CURR", "Investor currency{F}Devise investisseur",
GetType(String)), _

New CECubeColumn("INVESTOR_CLASS", "Investor type{F}Type investisseur",
GetType(String)), _

New CECubeColumn("INVESTOR_NAME", "Investor name{F}Nom investisseur",
GetType(String)), _

New CECubeColumn("INVESTMENT_ID", "ID investment{F}ID investissement",
GetType(String)), _
```

```
New CECubeColumn("MASTERINV_ID", "ID master investment{F}ID master  
investissement", GetType(String)), _  
  
New CECubeColumn("MASTERINV_NAME", "Name master investment{F}Nom master  
investissement", GetType(String)), _  
  
New CECubeColumn("BUYSHARES", "BuyShares", GetType(Double)), _  
  
New CECubeColumn("SELLSHARES", "SellShares", GetType(Double)), _  
  
New CECubeColumn("CURRENTSHARES", "CurrentShares", GetType(Double)), _  
  
New CECubeColumn("OWNEDSHARES", "OwnedShares", GetType(Double)), _  
  
New CECubeColumn("EXPPERCENTHOLDING", "ExpPercentHolding",  
GetType(Double)), _  
  
New CECubeColumn("EXPPERCENTHOLDINGFD", "ExpPercentHoldingFD",  
GetType(Double)), _  
  
New CECubeColumn("UNDRAWNCOMMITMENTINCURR",  
"UndrawnCommitmentInCurr", GetType(Double)), _  
  
New CECubeColumn("COMMITMENT", "Commitment", GetType(Double)), _  
  
New CECubeColumn("UNDRAWNCOMMITMENT", "Remaining  
Commitment{F}Engagement résiduel", GetType(Double)), _  
  
New CECubeColumn("INVESTMENT", "Total invested (cash out){F}Total investi (cash  
out)", GetType(Double)), _  
  
New CECubeColumn("INVEST", "Net invested{F}Net investi", GetType(Double)), _  
  
New CECubeColumn("OTHERAMOUNTCOST", "Buy fees{F}Frais d'achat",  
GetType(Double)), _  
  
New CECubeColumn("ALLOTHERCOST", "Other fees{F}Autres frais",  
GetType(Double)), _  
  
New CECubeColumn("EXIT", "Cash in{F}Cash in", GetType(Double)), _
```

New CECubeColumn("REIMBURSEMENT", "Proceeds amount{F}Montant cédé",  
GetType(Double)), \_

New CECubeColumn("SALESPROCEEDS", "Proceeds amount (equity){F}Montant cédé  
(action)", GetType(Double)), \_

New CECubeColumn("REPAYMENTS", "Proceeds amount (debt){F}Montant cédé  
(dettes)", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTINCOME", "Sale fees{F}Frais de ventes",  
GetType(Double)), \_

New CECubeColumn("ALLOTHERINCOME", "Total income{F}Total revenus",  
GetType(Double)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Double)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Double)), \_

New CECubeColumn("OTHERINCOME", "Other incomes{F}Autre revenus",  
GetType(Double)), \_

New CECubeColumn("REALIZEDCOST", "Cost of sale{F}Prix de revient cédé",  
GetType(Double)), \_

New CECubeColumn("INTERNALTRANS", "InternalTrans", GetType(Double)), \_

New CECubeColumn("VALUATION", "Valuation{F}Valorisation", GetType(Double)), \_

New CECubeColumn("ADJVALUATION", "Adjusted valuation{F}Valorisation ajustée",  
GetType(Double)), \_

New CECubeColumn("VALUATIONWOPROV", "ValuationWOProv", GetType(Double)),

—

New CECubeColumn("CURRENTCOST", "Current cost{F}Prix de revient",  
GetType(Double)), \_

New CECubeColumn("CAPITALISEDINTEREST", "CapitalisedInterest",  
GetType(Double)), \_

New CECubeColumn("REALIZEDGAINLOSS", "Realized gain/loss{F}+/- values réalisées", GetType(Double)), \_

New CECubeColumn("REALIZEDGLFX", "Realized GL (FX){F}+/- values réalisées (change)", GetType(Double)), \_

New CECubeColumn("REALIZEDGLINV", "Realized GL (market){F}+/- values réalisées (marché)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSS", "Unrealized gain/loss{F}+/- values non réalisées", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSFX", "Unrealized GL (FX){F}+/- values non réalisées (change)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSINV", "Unrealized GL (market){F}+/- values non réalisées (marché)", GetType(Double)), \_

New CECubeColumn("REALIZEDGL", "RealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGL", "UnRealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLFX", "UnRealizedGLFX (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLINV", "UnRealizedGLInv (deprecated)", GetType(Double)), \_

New CECubeColumn("COMMITTEDSHARES", "CommittedShares", GetType(Double)), \_

—

New CECubeColumn("UNDRAWNSHARES", "UndrawnShares", GetType(Double)), \_

New CECubeColumn("SHARESCALLEDINCOMMITMENT", "SharesCalledInCommitment", GetType(Double)), \_

New CECubeColumn("INVESTMENTINCOMMITMENT", "InvestmentInCommitment", GetType(Double)), \_

```
New CECubeColumn("WRITEOFFCOST", "WriteoffCost", GetType(Double)), __  
New CECubeColumn("PRINCIPAL", "Principal", GetType(Double)), __  
New CECubeColumn("BALANCE", "Balance", GetType(Double)) __  
New CECubeColumn("USEDFORIRR", "Used for IRR calculation", GetType(Boolean)) __  
New CECubeColumn("USEDFORNETIRR", "Used for net IRR calculation,  
GetType(Boolean)) __  
} __
```

## 6.8.2. InstrumentPositions

### 6.8.2.1. INSTRUMENTPOSITIONS

```
New CECubeColumn() { __  
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",  
GetType(DateTime)), __  
New CECubeColumn("CURRENCY", "Currency{F}Devise", GetType(String)), __  
New CECubeColumn("SECURITY_ID", "ID Investment Security{F}ID titre  
investissement", GetType(String)), __  
New CECubeColumn("SECURITY_NAME", "Security name{F}Nom du titre",  
GetType(String)), __  
New CECubeColumn("SECURITY_CURRENCY", "Security currency{F}Devise du titre",  
GetType(String)), __  
New CECubeColumn("SECURITY_STATUS", "Security status{F}Statut du titre",  
GetType(String)), __  
New CECubeColumn("SECURITY_COUNTRY", "Security country{F}Pays du titre",  
GetType(String)), __  
New CECubeColumn("SECURITY_CLASS_NAME", "Security class{F}Classe de titre",  
GetType(String)), __
```

```
New CECubeColumn("SECURITY_TYPE_CODE", "Type", GetType(String)), _  
New CECubeColumn("SECURITY_GROUPBY", "InstrumentGROUPBY",  
GetType(String)), _  
New CECubeColumn("SECURITY_ISIN", "Security ISIN code{F}Code ISIN du titre",  
GetType(String)), _  
New CECubeColumn("SECURITY_TICKER", "Security Ticker code{F}Code Ticker du  
titre", GetType(String)), _  
New CECubeColumn("INSTRUMENTFATHER_ID", "InstrumentFATHERID",  
GetType(String)), _  
New CECubeColumn("FATHERINSTRSTARTDATE", "fatherInstrSTARTDATE",  
GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRTERMDATE", "fatherInstrTERMDATE",  
GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRSTARTREPDATE", "fatherInstrSTARTREPDATE",  
GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRCLOSEREPDATE",  
"fatherInstrCLOSEREPDATE", GetType(DateTime)), _  
New CECubeColumn("FATHERINSTRMARKET", "fatherInstrMARKET",  
GetType(String)), _  
New CECubeColumn("FATHERINSTRBASERATE", "fatherInstrBASERATE",  
GetType(Double)), _  
New CECubeColumn("FATHERINSTRINTERESTRATE", "fatherInstrINTERESTRATE",  
GetType(Double)), _  
New CECubeColumn("FATHERINSTRCONVERSIONRATE",  
"fatherInstrCONVERSIONRATE", GetType(Double)), _  
New CECubeColumn("FATHERINSTRSTARTINTERESTDATE",  
"fatherInstrSTARTINTERESTDATE", GetType(DateTime)), _
```

```
New CECubeColumn("FATHERINSTRCLOSINGINTERESTDATE",
"fatherInstrCLOSINGINTERESTDATE", GetType(DateTime)), _

New CECubeColumn("PROPERTY_ID", "Property ID", GetType(String)), _

New CECubeColumn("COMPANY_ID", "ID Company{F}ID société", GetType(String)), _

New CECubeColumn("COMPANY_CURR", "Company currency{F}Devise société",
GetType(String)), _

New CECubeColumn("COMPANY_NAME", "Company name{F}Nom société",
GetType(String)), _

New CECubeColumn("COMPANY_COUNTRY", "Company country{F}Pays société",
GetType(String)), _

New CECubeColumn("COMPANY_INVESTTYPE", "Company investment type{F}Type
investissement société", GetType(String)), _

New CECubeColumn("COMPANY_STAGE", "Company stage{F}Devise société",
GetType(String)), _

New CECubeColumn("COMPANY_INSTRUMENTNAME",
"InvesteeINSTRUMENTNAME", GetType(String)), _

New CECubeColumn("INVESTOR_ID", "ID investor{F}ID investisseur",
GetType(String)), _

New CECubeColumn("INVESTOR_CURR", "Investor currency{F}Devise investisseur",
GetType(String)), _

New CECubeColumn("INVESTOR_CLASS", "Investor type{F}Type investisseur",
GetType(String)), _

New CECubeColumn("INVESTOR_NAME", "Investor name{F}Nom investisseur",
GetType(String)), _

New CECubeColumn("INVESTMENT_ID", "ID investment{F}ID investissement",
GetType(String)), _
```

New CECubeColumn("INVESTMENTEXITDATE", "Investment exit date{F}Date de sortie investissement", GetType(Date)), \_

New CECubeColumn("INVESTMENTISEXITED", "Investment exited ?{F}Investissement cédé ?", GetType(Boolean)), \_

New CECubeColumn("MASTERINV\_ID", "ID master investment{F}ID master investissement", GetType(String)), \_

New CECubeColumn("MASTERINV\_NAME", "Name master investment{F}Nom master investissement", GetType(String)), \_

New CECubeColumn("BUYSHARES", "BuyShares", GetType(Double)), \_

New CECubeColumn("SELLSHARES", "SellShares", GetType(Double)), \_

New CECubeColumn("CURRENTSHARES", "CurrentShares", GetType(Double)), \_

New CECubeColumn("OWNEDSHARES", "OwnedShares", GetType(Double)), \_

New CECubeColumn("OWNEDFDSHARES", "OwnedFDShares", GetType(Double)), \_

New CECubeColumn("COMPNBSHARES", "CompNbShares", GetType(Double)), \_

New CECubeColumn("COMPFDSHARES", "CompFDShares", GetType(Double)), \_

New CECubeColumn("NDSTAKE", "ND stake", GetType(Double)), \_

New CECubeColumn("FDSTAKE", "FD Stake", GetType(Double)), \_

New CECubeColumn("CURRENTFDSHARES", "CurrentFDShares", GetType(Double)), \_

-

New CECubeColumn("UNDRAWNCOMMITMENTINCURR",  
"UndrawnCommitmentInCurr", GetType(Double)), \_

New CECubeColumn("COMMITMENT", "Commitment", GetType(Double)), \_

New CECubeColumn("UNDRAWNCOMMITMENT", "Remaining Commitment{F}Engagement résiduel", GetType(Double)), \_

New CECubeColumn("INVESTMENT", "Total invested (cash out){F}Total investi (cash out)", GetType(Double)), \_

New CECubeColumn("INVEST", "Net invested{F}Net investi", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTCOST", "Buy fees{F}Frais d'achat", GetType(Double)), \_

New CECubeColumn("ALLOTHERCOST", "Other fees{F}Autres frais", GetType(Double)), \_

New CECubeColumn("EXIT", "Cash in{F}Cash in", GetType(Double)), \_

New CECubeColumn("REIMBURSEMENT", "Proceeds amount{F}Montant cédé", GetType(Double)), \_

New CECubeColumn("SALESPROCEEDS", "Proceeds amount (equity){F}Montant cédé (action)", GetType(Double)), \_

New CECubeColumn("REPAYMENTS", "Proceeds amount (debt){F}Montant cédé (dettes)", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTINCOME", "Sale fees{F}Frais de ventes", GetType(Double)), \_

New CECubeColumn("ALLOTHERINCOME", "Total income{F}Total revenus", GetType(Double)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Double)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Double)), \_

New CECubeColumn("OTHERINCOME", "Other incomes{F}Autre revenus", GetType(Double)), \_

New CECubeColumn("REALIZEDCOST", "Cost of sale{F}Prix de revient cédé", GetType(Double)), \_

New CECubeColumn("INTERNALTRANS", "InternalTrans", GetType(Double)), \_

New CECubeColumn("LASTVALUATION", "Last valuation{F}Dernière valorisation",  
GetType(Double)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last valuation date{F}Date de dernière  
valorisation", GetType(DateTime)), \_

New CECubeColumn("ADJVALUATION", "Adjusted valuation{F}Valorisation ajustée",  
GetType(Double)), \_

New CECubeColumn("VALUATIONWOPROV", "ValuationWOProv", GetType(Double)),

\_

New CECubeColumn("CURRENTCOST", "Current cost{F}Prix de revient",  
GetType(Double)), \_

New CECubeColumn("CAPITALISEDINTEREST", "CapitalisedInterest",  
GetType(Double)), \_

New CECubeColumn("REALIZEDGAINLOSS", "Realized gain/loss{F}+- values  
réalisées", GetType(Double)), \_

New CECubeColumn("REALIZEDGLFX", "Realized GL (FX){F}+- values réalisées  
(change)", GetType(Double)), \_

New CECubeColumn("REALIZEDGLINV", "Realized GL (market){F}+- values réalisées  
(marché)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSS", "Unrealized gain/loss{F}+- values  
non réalisées", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSFX", "Unrealized GL (FX){F}+- values  
non réalisées (change)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSINV", "Unrealized GL (market){F}+-  
values non réalisées (marché)", GetType(Double)), \_

New CECubeColumn("REALIZEDGL", "RealizedGL", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGL", "UnRealizedGL", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLFX", "UnRealizedGLFX", GetType(Double)), \_  
New CECubeColumn("UNREALIZEDGLINV", "UnRealizedGLInv", GetType(Double)), \_  
New CECubeColumn("COMMITTEDSHARES", "CommittedShares", GetType(Double)), \_  
—  
New CECubeColumn("UNDRAWNSHARES", "UndrawnShares", GetType(Double)), \_  
New CECubeColumn("SHARESCALLEDINCOMMITMENT",  
"SharesCalledInCommitment", GetType(Double)), \_  
New CECubeColumn("INVESTMENTINCOMMITMENT", "InvestmentInCommitment",  
GetType(Double)), \_  
New CECubeColumn("WRITEOFFCOST", "WriteoffCost", GetType(Double)), \_  
New CECubeColumn("PRINCIPAL", "Principal", GetType(Double)), \_  
New CECubeColumn("BALANCE", "Balance", GetType(Double)), \_  
New CECubeColumn("ACCRUALS", "Accruals", GetType(Double)), \_  
New CECubeColumn("PIK", "PIK", GetType(Double)), \_  
New CECubeColumn("DUEINTERESTS", "Due interests", GetType(Double)), \_  
New CECubeColumn("PAIDINTERESTS", "Paid interests", GetType(Double)), \_  
New CECubeColumn("CASHACCRUALS", "Cash accruals", GetType(Double)), \_  
New CECubeColumn("PIKACCRUALS", "PIK accruals", GetType(Double)), \_  
New CECubeColumn("CASHACTUALDAYS", "Cash actual days", GetType(Double)), \_  
New CECubeColumn("PIKACTUALDAYS", "PIK actual days", GetType(Double)), \_  
New CECubeColumn("TOTALVALUE", "Total value{F}Valeur totale", GetType(Double)), \_  
—  
New CECubeColumn("MULTIPLE", "Multiple", GetType(Double)), \_

```
New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _  
New CECubeColumn("IRRNONEXIT", "IRR Non Exit{F}TRI non cédé",  
GetType(Double)), _  
New CECubeColumn("IRREXIT", "IRR Exit{F}TRI cédé", GetType(Double)), _  
New CECubeColumn("NETIRR", "Net IRR{F}TRI net", GetType(Double)), _  
New CECubeColumn("NETIRRNONEXIT", "Net IRR Non Exit{F}TRI net non cédé",  
GetType(Double)), _  
New CECubeColumn("NETIRREXIT", "Net IRR Exit{F}TRI net cédé", GetType(Double)), _  
—  
New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",  
GetType(Double)), _  
New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _  
New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _  
New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _  
New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _  
New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _  
} _
```

 For calculated values (such as: Total\_Value, Expenses, Realized\_GainLoss, Interests, Dividends, Other\_Income, Valuation, LoanAccruals\_Principal, LoanAccruals\_PIK, LoanAccruals\_CashAccrual, LoanAccruals\_PiKAccrual, Multiple, IRR, Realized\_IRR, RealizedGainLossFX, UnrealizedGainLossFX), the currency taken into account is the one selected by USE\_INVESTEE\_CURR.

## 6.8.3. RefInstrumentPositions

### 6.8.3.1. REFINSTRUMENTPOSITIONS

```
New CECubeColumn() { _
```

```
    New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",  
        GetType(DateTime)), _
```

```
    New CECubeColumn("CURRENCY", "Currency{F}Devise", GetType(String)), _
```

```
    New CECubeColumn("SECURITY_ID", "ID Investment Security{F}ID titre  
        investissement", GetType(String)), _
```

```
    New CECubeColumn("SECURITY_NAME", "Security name{F}Nom du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_CURRENCY", "Security currency{F}Devise du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_STATUS", "Security status{F}Statut du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_COUNTRY", "Security country{F}Pays du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_CLASS_NAME", "Security class{F}Classe de titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_TYPE_CODE", "Type", GetType(String)), _
```

```
    New CECubeColumn("SECURITY_GROUPBY", "InstrumentGROUPBY",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_ISIN", "Security ISIN code{F}Code ISIN du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_TICKER", "Security Ticker code{F}Code Ticker du  
        titre", GetType(String)), _
```

```
New CECubeColumn("SECURITY_STARTDATE", "fatherInstrSTARTDATE",
GetType(DateTime)), _

New CECubeColumn("SECURITY_TERMDATE", "fatherInstrTERMDATE",
GetType(DateTime)), _

New CECubeColumn("SECURITY_STARTREPDATE", "fatherInstrSTARTREPDATE",
GetType(DateTime)), _

New CECubeColumn("SECURITY_CLOSEREPDATE", "fatherInstrCLOSEREPDATE",
GetType(DateTime)), _

New CECubeColumn("SECURITY_MARKET", "fatherInstrMARKET", GetType(String)),
_

New CECubeColumn("SECURITY_BASERATE", "fatherInstrBASERATE",
GetType(Double)), _

New CECubeColumn("SECURITY_INTERESTRATE", "fatherInstrINTERESTRATE",
GetType(Double)), _

New CECubeColumn("SECURITY_CONVERSIONRATE",
"fatherInstrCONVERSIONRATE", GetType(Double)), _

New CECubeColumn("SECURITY_STARTINTERESTDATE",
"fatherInstrSTARTINTERESTDATE", GetType(DateTime)), _

New CECubeColumn("SECURITY_CLOSINGINTERESTDATE",
"fatherInstrCLOSINGINTERESTDATE", GetType(DateTime)), _

New CECubeColumn("PROPERTY_ID", "Property ID", GetType(String)), _

New CECubeColumn("COMPANY_ID", "ID Company{F}ID société", GetType(String)), _

New CECubeColumn("COMPANY_CURR", "Company currency{F}Devise société",
GetType(String)), _

New CECubeColumn("COMPANY_NAME", "Company name{F}Nom société",
GetType(String)), _
```

```
New CECubeColumn("COMPANY_COUNTRY", "Company country{F}Pays société",
GetType(String)), _

New CECubeColumn("COMPANY_INVESTTYPE", "Company investment type{F}Type
investissement société", GetType(String)), _

New CECubeColumn("COMPANY_STAGE", "Company stage{F}Devise société",
GetType(String)), _

New CECubeColumn("BUYSHARES", "BuyShares", GetType(Double)), _

New CECubeColumn("SELLSHARES", "SellShares", GetType(Double)), _

New CECubeColumn("CURRENTSHARES", "CurrentShares", GetType(Double)), _

New CECubeColumn("COMPNBSHARES", "CompNbShares", GetType(Double)), _

New CECubeColumn("COMPFDSHARES", "CompFDShares", GetType(Double)), _

New CECubeColumn("CURRENTFDSHARES", "CurrentFDShares", GetType(Double)),
-

New CECubeColumn("UNDRAWNCOMMITMENTINCURR",
"UndrawnCommitmentInCurr", GetType(Double)), _

New CECubeColumn("COMMITMENT", "Commitment", GetType(Double)), _

New CECubeColumn("UNDRAWNCOMMITMENT", "Remaining
Commitment{F}Engagement résiduel", GetType(Double)), _

New CECubeColumn("INVESTMENT", "Total invested (cash out){F}Total investi (cash
out)", GetType(Double)), _

New CECubeColumn("INVEST", "Net invested{F}Net investi", GetType(Double)), _

New CECubeColumn("OTHERAMOUNTCOST", "Buy fees{F}Frais d'achat",
GetType(Double)), _

New CECubeColumn("ALLOTHERCOST", "Other fees{F}Autres frais",
GetType(Double)), _
```

New CECubeColumn("EXIT", "Cash in{F}Cash in", GetType(Double)), \_

New CECubeColumn("REIMBURSEMENT", "Proceeds amount{F}Montant cédé", GetType(Double)), \_

New CECubeColumn("SALESPROCEEDS", "Proceeds amount (equity){F}Montant cédé (action)", GetType(Double)), \_

New CECubeColumn("REPAYMENTS", "Proceeds amount (debt){F}Montant cédé (dette)", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTINCOME", "Sale fees{F}Frais de ventes", GetType(Double)), \_

New CECubeColumn("ALLOTHERINCOME", "Total income{F}Total revenus", GetType(Double)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Double)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Double)), \_

New CECubeColumn("OTHERINCOME", "Other incomes{F}Autre revenus", GetType(Double)), \_

New CECubeColumn("REALIZEDCOST", "Cost of sale{F}Prix de revient cédé", GetType(Double)), \_

New CECubeColumn("INTERNALTRANS", "InternalTrans", GetType(Double)), \_

New CECubeColumn("LASTVALUATION", "Last valuation{F}Dernière valorisation", GetType(Double)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last valuation date{F}Date de dernière valorisation", GetType(DateTime)), \_

New CECubeColumn("ADJVALUATION", "Adjusted valuation{F}Valorisation ajustée", GetType(Double)), \_

New CECubeColumn("VALUATIONWOPROV", "ValuationWOProv", GetType(Double)), \_

New CECubeColumn("CURRENTCOST", "Current cost{F}Prix de revient",  
GetType(Double)), \_

New CECubeColumn("CAPITALISEDINTEREST", "CapitalisedInterest",  
GetType(Double)), \_

New CECubeColumn("REALIZEDGAINLOSS", "Realized gain/loss{F}+/- values  
réalisées", GetType(Double)), \_

New CECubeColumn("REALIZEDGLFX", "Realized GL (FX){F}+/- values réalisées  
(change)", GetType(Double)), \_

New CECubeColumn("REALIZEDGLINV", "Realized GL (market){F}+/- values réalisées  
(marché)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSS", "Unrealized gain/loss{F}+/- values  
non réalisées", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSFX", "Unrealized GL (FX){F}+/- values  
non réalisées (change)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSINV", "Unrealized GL (market){F}+/-  
values non réalisées (marché)", GetType(Double)), \_

New CECubeColumn("REALIZEDGL", "RealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGL", "UnRealizedGL (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLFX", "UnRealizedGLFX (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLINV", "UnRealizedGLInv (deprecated)",  
GetType(Double)), \_

New CECubeColumn("COMMITTEDSHARES", "CommittedShares", GetType(Double)),

—

New CECubeColumn("UNDRAWNSHARES", "UndrawnShares", GetType(Double)), \_

```
New CECubeColumn("SHARESCALLEDINCOMMITMENT",
"SharesCalledInCommitment", GetType(Double)), _

New CECubeColumn("INVESTMENTINCOMMITMENT", "InvestmentInCommitment",
GetType(Double)), _

New CECubeColumn("WRITEOFFCOST", "WriteoffCost", GetType(Double)), _

New CECubeColumn("PRINCIPAL", "Principal", GetType(Double)), _

New CECubeColumn("BALANCE", "Balance", GetType(Double)), _

New CECubeColumn("ACCRUALS", "Accruals", GetType(Double)), _

New CECubeColumn("PIK", "PIK ", GetType(Double)), _

New CECubeColumn("DUEINTERESTS", "Due interests", GetType(Double)), _

New CECubeColumn("REALDUEINTERESTS", "Real Due interests", GetType(Double)),
-

New CECubeColumn("PAIDINTERESTS", "Paid interests", GetType(Double)), _

New CECubeColumn("CASHACCRUALS", "Cash accruals", GetType(Double)), _

New CECubeColumn("PIKACCRUALS", "PIK accruals", GetType(Double)), _

New CECubeColumn("CASHACTUALDAYS", "Cash actual days", GetType(Double)), _

New CECubeColumn("PIKACTUALDAYS", "PIK actual days", GetType(Double)), _

New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _

New CECubeColumn("IRRNONEEXIT", "IRR Non Exit{F}TRI non cédé",
GetType(Double)), _

New CECubeColumn("IRREXIT", "IRR Exit{F}TRI cédé", GetType(Double)), _

New CECubeColumn("NETIRR", "Net IRR{F}TRI net", GetType(Double)), _
```

```
New CECubeColumn("NETIRRNONEXIT", "Net IRR Non Exit{F}TRI net non cédé",
GetType(Double)), _

New CECubeColumn("NETIRREXIT", "Net IRR Exit{F}TRI net cédé", GetType(Double)),
_

New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",
GetType(Double)), _

New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _

New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _

New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _

New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _

New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _

}
```

## 6.8.4. InvestmentPositions

### 6.8.4.1. INVESTMENTPOSITIONS

```
New CECubeColumn() { _

New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",
GetType(DateTime)), _

New CECubeColumn("CURRENCY", "Currency{F}Devise", GetType(String)), _

New CECubeColumn("INVESTMENT_ID", "ID investment{F}ID investissement",
GetType(String)), _

New CECubeColumn("INVESTMENTEXITDATE", "Investment exit date{F}Date de sortie
investissement", GetType(Date)), _

New CECubeColumn("INVESTMENTISEXITED", "Investment exited ?{F}Investissement
cédé ?", GetType(Boolean)), _
```

```
New CECubeColumn("PROPERTY_ID", "Property ID", GetType(String)), _  
New CECubeColumn("COMPANY_ID", "ID Company{F}ID société", GetType(String)), _  
New CECubeColumn("COMPANY_CURR", "Company currency{F}Devise société",  
GetType(String)), _  
New CECubeColumn("COMPANY_NAME", "Company name{F}Nom société",  
GetType(String)), _  
New CECubeColumn("COMPANY_COUNTRY", "Company country{F}Pays société",  
GetType(String)), _  
New CECubeColumn("COMPANY_INVESTTYPE", "Company investment type{F}Type  
investissement société", GetType(String)), _  
New CECubeColumn("COMPANY_STAGE", "Company stage{F}Devise société",  
GetType(String)), _  
New CECubeColumn("INVESTOR_ID", "ID investor{F}ID investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTOR_CURR", "Investor currency{F}Devise investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTOR_CLASS", "Investor type{F}Type investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTOR_NAME", "Investor name{F}Nom investisseur",  
GetType(String)), _  
New CECubeColumn("MASTERINV_ID", "ID master investment{F}ID master  
investissement", GetType(String)), _  
New CECubeColumn("MASTERINV_NAME", "Name master investment{F}Nom master  
investissement", GetType(String)), _  
New CECubeColumn("BUYSHARES", "BuyShares", GetType(Double)), _  
New CECubeColumn("SELLSHARES", "SellShares", GetType(Double)), _
```

New CECubeColumn("CURRENTSHARES", "CurrentShares", GetType(Double)), \_  
New CECubeColumn("OWNEDSHARES", "OwnedShares", GetType(Double)), \_  
New CECubeColumn("OWNEDFDSHARES", "OwnedFDShares", GetType(Double)), \_  
New CECubeColumn("COMPNBSHARES", "CompNbShares", GetType(Double)), \_  
New CECubeColumn("COMPFDSHARES", "CompFDShares", GetType(Double)), \_  
New CECubeColumn("NDSTAKE", "ND stake", GetType(Double)), \_  
New CECubeColumn("FDSTAKE", "FD Stake", GetType(Double)), \_  
New CECubeColumn("CURRENTFDSHARES", "CurrentFDShares", GetType(Double)), \_  
—  
New CECubeColumn("UNDRAWNCOMMITMENTINCURR",  
"UndrawnCommitmentInCurr", GetType(Double)), \_  
New CECubeColumn("COMMITMENT", "Commitment", GetType(Double)), \_  
New CECubeColumn("UNDRAWNCOMMITMENT", "Remaining  
Commitment{F}Engagement résiduel", GetType(Double)), \_  
New CECubeColumn("INVESTMENT", "Total invested (cash out){F}Total investi (cash  
out)", GetType(Double)), \_  
New CECubeColumn("INVEST", "Net invested{F}Net investi", GetType(Double)), \_  
New CECubeColumn("OTHERAMOUNTCOST", "Buy fees{F}Frais d'achat",  
GetType(Double)), \_  
New CECubeColumn("ALLOTHERCOST", "Other fees{F}Autres frais",  
GetType(Double)), \_  
New CECubeColumn("EXIT", "Cash in{F}Cash in", GetType(Double)), \_  
New CECubeColumn("REIMBURSEMENT", "Proceeds amount{F}Montant cédé",  
GetType(Double)), \_

New CECubeColumn("SALESPROCEEDS", "Proceeds amount (equity){F}Montant cédé (action)", GetType(Double)), \_

New CECubeColumn("REPAYMENTS", "Proceeds amount (debt){F}Montant cédé (dette)", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTINCOME", "Sale fees{F}Frais de ventes", GetType(Double)), \_

New CECubeColumn("ALLOTHERINCOME", "Total income{F}Total revenus", GetType(Double)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Double)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Double)), \_

New CECubeColumn("OTHERINCOME", "Other incomes{F}Autre revenus", GetType(Double)), \_

New CECubeColumn("REALIZEDCOST", "Cost of sale{F}Prix de revient cédé", GetType(Double)), \_

New CECubeColumn("INTERNALTRANS", "InternalTrans", GetType(Double)), \_

New CECubeColumn("LASTVALUATION", "Last valuation{F}Dernière valorisation", GetType(Double)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last valuation date{F}Date de dernière valorisation", GetType(DateTime)), \_

New CECubeColumn("ADJVALUATION", "Adjusted valuation{F}Valorisation ajustée", GetType(Double)), \_

New CECubeColumn("VALUATIONWOPROV", "ValuationWOProv", GetType(Double)), \_

—

New CECubeColumn("CURRENTCOST", "Current cost{F}Prix de revient", GetType(Double)), \_

New CECubeColumn("CAPITALISEDINTEREST", "CapitalisedInterest",  
GetType(Double)), \_

New CECubeColumn("REALIZEDGAINLOSS", "Realized gain/loss{F}+/- values  
réalisées", GetType(Double)), \_

New CECubeColumn("REALIZEDGLFX", "Realized GL (FX){F}+/- values réalisées  
(change)", GetType(Double)), \_

New CECubeColumn("REALIZEDGLINV", "Realized GL (market){F}+/- values réalisées  
(marché)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSS", "Unrealized gain/loss{F}+/- values  
non réalisées", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSFX", "Unrealized GL (FX){F}+/- values  
non réalisées (change)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSINV", "Unrealized GL (market){F}+/-  
values non réalisées (marché)", GetType(Double)), \_

New CECubeColumn("REALIZEDGL", "RealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGL", "UnRealizedGL (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLFX", "UnRealizedGLFX (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLINV", "UnRealizedGLInv (deprecated)",  
GetType(Double)), \_

New CECubeColumn("COMMITTEDSHARES", "CommittedShares", GetType(Double)),

—

New CECubeColumn("UNDRAWNSHARES", "UndrawnShares", GetType(Double)), \_

New CECubeColumn("SHARESCALLEDINCOMMITMENT",  
"SharesCalledInCommitment", GetType(Double)), \_

```
New CECubeColumn("INVESTMENTINCOMMITMENT", "InvestmentInCommitment",
GetType(Double)), _

New CECubeColumn("WRITEOFFCOST", "WriteoffCost", GetType(Double)), _

New CECubeColumn("PRINCIPAL", "Principal", GetType(Double)), _

New CECubeColumn("BALANCE", "Balance", GetType(Double)), _

New CECubeColumn("ACCRUALS", "Accruals", GetType(Double)), _

New CECubeColumn("PIK", "PIK ", GetType(Double)), _

New CECubeColumn("DUEINTERESTS", "Due interests", GetType(Double)), _

New CECubeColumn("REALDUEINTERESTS", "Real Due interests", GetType(Double)),

—

New CECubeColumn("PAIDINTERESTS", "Paid interests", GetType(Double)), _

New CECubeColumn("CASHACCRUALS", "Cash accruals", GetType(Double)), _

New CECubeColumn("PIKACCRUALS", "PIK accruals", GetType(Double)), _

New CECubeColumn("CASHACTUALDAYS", "Cash actual days", GetType(Double)), _

New CECubeColumn("PIKACTUALDAYS", "PIK actual days", GetType(Double)), _

New CECubeColumn("TOTALVALUE", "Total value{F}Valeur totale", GetType(Double)),

—

New CECubeColumn("MULTIPLE", "Multiple", GetType(Double)), _

New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _

New CECubeColumn("IRRNONEEXIT", "IRR Non Exit{F}TRI non cédé",
GetType(Double)), _

New CECubeColumn("IRREXIT", "IRR Exit{F}TRI cédé", GetType(Double)), _

New CECubeColumn("NETIRR", "Net IRR{F}TRI net", GetType(Double)), _
```

```
New CECubeColumn("NETIRRNONEXIT", "Net IRR Non Exit{F}TRI net non cédé",
GetType(Double)), _

New CECubeColumn("NETIRREXIT", "Net IRR Exit{F}TRI net cédé", GetType(Double)),
_

New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",
GetType(Double)), _

New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _

New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _

New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _

New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _

New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _

}
```



For calculated values (such as: Total\_Value, Expenses, Realized\_GainLoss, Interests, Dividends, Other\_Income, Valuation, LoanAccruals\_Principal, LoanAccruals\_PIK, LoanAccruals\_CashAccrual, LoanAccruals\_PiKAccrual, Multiple, IRR, Realized\_IRR, RealizedGainLossFX, UnrealizedGainLossFX), the currency taken into account is the one selected by USE\_INVESTEE\_CURR.

## 6.8.5. InvestorPositions

### 6.8.5.1. INVESTORPOSITIONS

```
New CECubeColumn() { _

New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",
GetType(DateTime)), _

New CECubeColumn("CURRENCY", "Currency{F}Devise", GetType(String)), _

New CECubeColumn("INVESTOR_ID", "ID investor{F}ID investisseur",
GetType(String)), _
```

New CECubeColumn("INVESTOR\_CURR", "Investor currency{F}Devise investisseur", GetType(String)), \_

New CECubeColumn("INVESTOR\_CLASS", "Investor type{F}Type investisseur", GetType(String)), \_

New CECubeColumn("INVESTOR\_NAME", "Investor name{F}Nom investisseur", GetType(String)), \_

New CECubeColumn("MASTERINV\_ID", "ID master investment{F}ID master investissement", GetType(String)), \_

New CECubeColumn("MASTERINV\_NAME", "Name master investment{F}Nom master investissement", GetType(String)), \_

New CECubeColumn("UNDRAWNCOMMITMENTINCURR", "UndrawnCommitmentInCurr", GetType(Double)), \_

New CECubeColumn("COMMITMENT", "Commitment", GetType(Double)), \_

New CECubeColumn("UNDRAWNCOMMITMENT", "Remaining Commitment{F}Engagement résiduel", GetType(Double)), \_

New CECubeColumn("INVESTMENT", "Total invested (cash out){F}Total investi (cash out)", GetType(Double)), \_

New CECubeColumn("INVEST", "Net invested{F}Net investi", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTCOST", "Buy fees{F}Frais d'achat", GetType(Double)), \_

New CECubeColumn("ALLOTHERCOST", "Other fees{F}Autres frais", GetType(Double)), \_

New CECubeColumn("EXIT", "Cash in{F}Cash in", GetType(Double)), \_

New CECubeColumn("REIMBURSEMENT", "Proceeds amount{F}Montant cédé", GetType(Double)), \_

New CECubeColumn("SALESPROCEEDS", "Proceeds amount (equity){F}Montant cédé (action)", GetType(Double)), \_

New CECubeColumn("REPAYMENTS", "Proceeds amount (debt){F}Montant cédé (dette)", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTINCOME", "Sale fees{F}Frais de ventes", GetType(Double)), \_

New CECubeColumn("ALLOTHERINCOME", "Total income{F}Total revenus", GetType(Double)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Double)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Double)), \_

New CECubeColumn("OTHERINCOME", "Other incomes{F}Autre revenus", GetType(Double)), \_

New CECubeColumn("REALIZEDCOST", "Cost of sale{F}Prix de revient cédé", GetType(Double)), \_

New CECubeColumn("INTERNALTRANS", "InternalTrans", GetType(Double)), \_

New CECubeColumn("LASTVALUATION", "Last valuation{F}Dernière valorisation", GetType(Double)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last valuation date{F}Date de dernière valorisation", GetType(DateTime)), \_

New CECubeColumn("ADJVALUATION", "Adjusted valuation{F}Valorisation ajustée", GetType(Double)), \_

New CECubeColumn("VALUATIONWOPROV", "ValuationWOProv", GetType(Double)), \_

—

New CECubeColumn("CURRENTCOST", "Current cost{F}Prix de revient", GetType(Double)), \_

New CECubeColumn("CAPITALISEDINTEREST", "CapitalisedInterest",  
GetType(Double)), \_

New CECubeColumn("REALIZEDGAINLOSS", "Realized gain/loss{F}+/- values  
réalisées", GetType(Double)), \_

New CECubeColumn("REALIZEDGLFX", "Realized GL (FX){F}+/- values réalisées  
(change)", GetType(Double)), \_

New CECubeColumn("REALIZEDGLINV", "Realized GL (market){F}+/- values réalisées  
(marché)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSS", "Unrealized gain/loss{F}+/- values  
non réalisées", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSFX", "Unrealized GL (FX){F}+/- values  
non réalisées (change)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSINV", "Unrealized GL (market){F}+/-  
values non réalisées (marché)", GetType(Double)), \_

New CECubeColumn("REALIZEDGL", "RealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGL", "UnRealizedGL (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLFX", "UnRealizedGLFX (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLINV", "UnRealizedGLInv (deprecated)",  
GetType(Double)), \_

New CECubeColumn("COMMITTEDSHARES", "CommittedShares", GetType(Double)),

—

New CECubeColumn("UNDRAWNSHARES", "UndrawnShares", GetType(Double)), \_

New CECubeColumn("SHARESCALLEDINCOMMITMENT",  
"SharesCalledInCommitment", GetType(Double)), \_

```
New CECubeColumn("INVESTMENTINCOMMITMENT", "InvestmentInCommitment",
GetType(Double)), _

New CECubeColumn("WRITEOFFCOST", "WriteoffCost", GetType(Double)), _

New CECubeColumn("PRINCIPAL", "Principal", GetType(Double)), _

New CECubeColumn("BALANCE", "Balance", GetType(Double)), _

New CECubeColumn("ACCRUALS", "Accruals", GetType(Double)), _

New CECubeColumn("PIK", "PIK ", GetType(Double)), _

New CECubeColumn("DUEINTERESTS", "Due interests", GetType(Double)), _

New CECubeColumn("REALDUEINTERESTS", "Real Due interests", GetType(Double)),

—

New CECubeColumn("PAIDINTERESTS", "Paid interests", GetType(Double)), _

New CECubeColumn("CASHACCRUALS", "Cash accruals", GetType(Double)), _

New CECubeColumn("PIKACCRUALS", "PIK accruals", GetType(Double)), _

New CECubeColumn("CASHACTUALDAYS", "Cash actual days", GetType(Double)), _

New CECubeColumn("PIKACTUALDAYS", "PIK actual days", GetType(Double)), _

New CECubeColumn("TOTALVALUE", "Total value{F}Valeur totale", GetType(Double)),

—

New CECubeColumn("MULTIPLE", "Multiple", GetType(Double)), _

New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _

New CECubeColumn("IRRNONEEXIT", "IRR Non Exit{F}TRI non cédé",
GetType(Double)), _

New CECubeColumn("IRREXIT", "IRR Exit{F}TRI cédé", GetType(Double)), _

New CECubeColumn("NETIRR", "Net IRR{F}TRI net", GetType(Double)), _
```

```
New CECubeColumn("NETIRRNONEXIT", "Net IRR Non Exit{F}TRI net non cédé",  
GetType(Double)), _
```

```
New CECubeColumn("NETIRREXIT", "Net IRR Exit{F}TRI net cédé", GetType(Double)),
```

```
_
```

```
New CECubeColumn("EXITPORTFOLIOIRR", "IRR of realized portfolio {F}TRI du  
portefeuille cédé", GetType(Double)), _
```

```
New CECubeColumn("NONEXITPORTFOLIOIRR", "IRR of unrealized portfolio {F}TRI  
du portefeuille non cédé", GetType(Double)), _
```

```
New CECubeColumn("NONEXITPORTFOLIOIRRNONEXIT", "IRR non exit of  
unrealized portfolio {F}TRI non cédé du portefeuille non cédé", GetType(Double)), _
```

```
New CECubeColumn("NONEXITPORTFOLIOIRREXIT", "IRR exit of unrealized portfolio  
{F}TRI cédé du portefeuille non cédé", GetType(Double)), _
```

```
New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",  
GetType(Double)), _
```

```
New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _
```

```
New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _
```

```
New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _
```

```
New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _
```

```
New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _
```

```
} _
```

 For calculated values (such as: Total\_Value, Expenses, Realized\_GainLoss, Interests, Dividends, Other\_Income, Valuation, LoanAccruals\_Principal, LoanAccruals\_PIK, LoanAccruals\_CashAccrual, LoanAccruals\_PiKAccrual, Multiple, IRR, Realized\_IRR, RealizedGainLossFX, UnrealizedGainLossFX), the currency taken into account is the one selected by USE\_INVESTEE\_CURR.

## 6.8.6. InstrumentRounds

### 6.8.6.1. INSTRUMENTROUNDS

```
New CECubeColumn() { _
```

```
    New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",  
        GetType(DateTime)), _
```

```
    New CECubeColumn("CURRENCY", "Currency{F}Devise", GetType(String)), _
```

```
    New CECubeColumn("SECURITY_ID", "ID Investment Security{F}ID titre  
investissement", GetType(String)), _
```

```
    New CECubeColumn("SECURITY_NAME", "Security name{F}Nom du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("ROUND_ID", "ID Security Round{F}ID tranche",  
        GetType(String)), _
```

```
    New CECubeColumn("ROUND_NAME", "Security round name{F}Nom de la tranche",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_CURRENCY", "Security currency{F}Devise du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_STATUS", "Security status{F}Statut du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_COUNTRY", "Security country{F}Pays du titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_CLASS_NAME", "Security class{F}Classe de titre",  
        GetType(String)), _
```

```
    New CECubeColumn("SECURITY_TYPE_CODE", "Type", GetType(String)), _
```

```
    New CECubeColumn("SECURITY_GROUPBY", "InstrumentGROUPBY",  
        GetType(String)), _
```

```
New CECubeColumn("SECURITY_ISIN", "Security ISIN code{F}Code ISIN du titre",  
GetType(String)), _
```

```
New CECubeColumn("SECURITY_TICKER", "Security Ticker code{F}Code Ticker du  
titre", GetType(String)), _
```

```
New CECubeColumn("INSTRUMENTFATHER_ID", "InstrumentFATHERID",  
GetType(String)), _
```

```
New CECubeColumn("FATHERINSTRSTARTDATE", "fatherInstrSTARTDATE",  
GetType(DateTime)), _
```

```
New CECubeColumn("FATHERINSTRTERMDATE", "fatherInstrTERMDATE",  
GetType(DateTime)), _
```

```
New CECubeColumn("FATHERINSTRSTARTREPDATE", "fatherInstrSTARTREPDATE",  
GetType(DateTime)), _
```

```
New CECubeColumn("FATHERINSTRCLOSEREPDATE",  
"fatherInstrCLOSEREPDATE", GetType(DateTime)), _
```

```
New CECubeColumn("FATHERINSTRMARKET", "fatherInstrMARKET",  
GetType(String)), _
```

```
New CECubeColumn("FATHERINSTRBASERATE", "fatherInstrBASERATE",  
GetType(Double)), _
```

```
New CECubeColumn("FATHERINSTRINTERESTRATE", "fatherInstrINTERESTRATE",  
GetType(Double)), _
```

```
New CECubeColumn("FATHERINSTRCONVERSIONRATE",  
"fatherInstrCONVERSIONRATE", GetType(Double)), _
```

```
New CECubeColumn("FATHERINSTRSTARTINTERESTDATE",  
"fatherInstrSTARTINTERESTDATE", GetType(DateTime)), _
```

```
New CECubeColumn("FATHERINSTRCLOSINGINTERESTDATE",  
"fatherInstrCLOSINGINTERESTDATE", GetType(DateTime)), _
```

```
New CECubeColumn("PROPERTY_ID", "Property ID", GetType(String)), _
```

```
New CECubeColumn("COMPANY_ID", "ID Company{F}ID société", GetType(String)), _  
New CECubeColumn("COMPANY_CURR", "Company currency{F}Devise société",  
GetType(String)), _  
New CECubeColumn("COMPANY_NAME", "Company name{F}Nom société",  
GetType(String)), _  
New CECubeColumn("COMPANY_COUNTRY", "Company country{F}Pays société",  
GetType(String)), _  
New CECubeColumn("COMPANY_INVESTTYPE", "Company investment type{F}Type  
investissement société", GetType(String)), _  
New CECubeColumn("COMPANY_STAGE", "Company stage{F}Devise société",  
GetType(String)), _  
New CECubeColumn("COMPANY_INSTRUMENTNAME",  
"InvesteeINSTRUMENTNAME", GetType(String)), _  
New CECubeColumn("INVESTOR_ID", "ID investor{F}ID investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTOR_CURR", "Investor currency{F}Devise investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTOR_CLASS", "Investor type{F}Type investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTOR_NAME", "Investor name{F}Nom investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTMENT_ID", "ID investment{F}ID investissement",  
GetType(String)), _  
New CECubeColumn("MASTERINV_ID", "ID master investment{F}ID master  
investissement", GetType(String)), _  
New CECubeColumn("MASTERINV_NAME", "Name master investment{F}Nom master  
investissement", GetType(String)), _
```

New CECubeColumn("BUYSHARES", "BuyShares", GetType(Double)), \_  
New CECubeColumn("SELLSHARES", "SellShares", GetType(Double)), \_  
New CECubeColumn("CURRENTSHARES", "CurrentShares", GetType(Double)), \_  
New CECubeColumn("OWNEDSHARES", "OwnedShares", GetType(Double)), \_  
New CECubeColumn("OWNEDFDSHARES", "OwnedFDShares", GetType(Double)), \_  
New CECubeColumn("COMPNBSHARES", "CompNbShares", GetType(Double)), \_  
New CECubeColumn("COMPFDSHARES", "CompFDShares", GetType(Double)), \_  
New CECubeColumn("NDSTAKE", "ND stake", GetType(Double)), \_  
New CECubeColumn("FDSTAKE", "FD Stake", GetType(Double)), \_  
New CECubeColumn("CURRENTFDSHARES", "CurrentFDShares", GetType(Double)), \_  
—  
New CECubeColumn("UNDRAWNCOMMITMENTINCURR",  
"UndrawnCommitmentInCurr", GetType(Double)), \_  
New CECubeColumn("COMMITMENT", "Commitment", GetType(Double)), \_  
New CECubeColumn("UNDRAWNCOMMITMENT", "Remaining  
Commitment{F}Engagement résiduel", GetType(Double)), \_  
New CECubeColumn("INVESTMENT", "Total invested (cash out){F}Total investi (cash  
out)", GetType(Double)), \_  
New CECubeColumn("INVEST", "Net invested{F}Net investi", GetType(Double)), \_  
New CECubeColumn("OTHERAMOUNTCOST", "Buy fees{F}Frais d'achat",  
GetType(Double)), \_  
New CECubeColumn("ALLOTHERCOST", "Other fees{F}Autres frais",  
GetType(Double)), \_  
New CECubeColumn("EXIT", "Cash in{F}Cash in", GetType(Double)), \_

New CECubeColumn("REIMBURSEMENT", "Proceeds amount{F}Montant cédé",  
GetType(Double)), \_

New CECubeColumn("SALESPROCEEDS", "Proceeds amount (equity){F}Montant cédé  
(action)", GetType(Double)), \_

New CECubeColumn("REPAYMENTS", "Proceeds amount (debt){F}Montant cédé  
(dette)", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTINCOME", "Sale fees{F}Frais de ventes",  
GetType(Double)), \_

New CECubeColumn("ALLOTHERINCOME", "Total income{F}Total revenus",  
GetType(Double)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Double)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Double)), \_

New CECubeColumn("OTHERINCOME", "Other incomes{F}Autre revenus",  
GetType(Double)), \_

New CECubeColumn("REALIZEDCOST", "Cost of sale{F}Prix de revient cédé",  
GetType(Double)), \_

New CECubeColumn("INTERNALTRANS", "InternalTrans", GetType(Double)), \_

New CECubeColumn("LASTVALUATION", "Last valuation{F}Dernière valorisation",  
GetType(Double)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last valuation date{F}Date de dernière  
valorisation", GetType(DateTime)), \_

New CECubeColumn("ADJVALUATION", "Adjusted valuation{F}Valorisation ajustée",  
GetType(Double)), \_

New CECubeColumn("VALUATIONWOPROV", "ValuationWOProv", GetType(Double)),

-

New CECubeColumn("CURRENTCOST", "Current cost{F}Prix de revient",  
GetType(Double)), \_

New CECubeColumn("CAPITALISEDINTEREST", "CapitalisedInterest",  
GetType(Double)), \_

New CECubeColumn("REALIZEDGAINLOSS", "Realized gain/loss{F}+/- values  
réalisées", GetType(Double)), \_

New CECubeColumn("REALIZEDGLFX", "Realized GL (FX){F}+/- values réalisées  
(change)", GetType(Double)), \_

New CECubeColumn("REALIZEDGLINV", "Realized GL (market){F}+/- values réalisées  
(marché)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSS", "Unrealized gain/loss{F}+/- values  
non réalisées", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSFX", "Unrealized GL (FX){F}+/- values  
non réalisées (change)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSINV", "Unrealized GL (market){F}+/-  
values non réalisées (marché)", GetType(Double)), \_

New CECubeColumn("REALIZEDGL", "RealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGL", "UnRealizedGL (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLFX", "UnRealizedGLFX (deprecated)",  
GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLINV", "UnRealizedGLInv (deprecated)",  
GetType(Double)), \_

New CECubeColumn("COMMITTEDSHARES", "CommittedShares", GetType(Double)),

—

New CECubeColumn("UNDRAWNSHARES", "UndrawnShares", GetType(Double)), \_

```
New CECubeColumn("SHARESCALLEDINCOMMITMENT",
"SharesCalledInCommitment", GetType(Double)), _

New CECubeColumn("INVESTMENTINCOMMITMENT", "InvestmentInCommitment",
GetType(Double)), _

New CECubeColumn("WRITEOFFCOST", "WriteoffCost", GetType(Double)), _

New CECubeColumn("PRINCIPAL", "Principal", GetType(Double)), _

New CECubeColumn("BALANCE", "Balance", GetType(Double)), _

New CECubeColumn("ACCRUALS", "Accruals", GetType(Double)), _

New CECubeColumn("PIK", "PIK ", GetType(Double)), _

New CECubeColumn("DUEINTERESTS", "Due interests", GetType(Double)), _

New CECubeColumn("PAIDINTERESTS", "Paid interests", GetType(Double)), _

New CECubeColumn("CASHACCRUALS", "Cash accruals", GetType(Double)), _

New CECubeColumn("PIKACCRUALS", "PIK accruals", GetType(Double)), _

New CECubeColumn("CASHACTUALDAYS", "Cash actual days", GetType(Double)), _

New CECubeColumn("PIKACTUALDAYS", "PIK actual days", GetType(Double)), _

New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _

New CECubeColumn("IRRNONEXIT", "IRR Non Exit{F}TRI non cédé",
GetType(Double)), _

New CECubeColumn("IRREXIT", "IRR Exit{F}TRI cédé", GetType(Double)), _

New CECubeColumn("NETIRR", "Net IRR{F}TRI net", GetType(Double)), _

New CECubeColumn("NETIRRNONEXIT", "Net IRR Non Exit{F}TRI net non cédé",
GetType(Double)), _
```

```
New CECubeColumn("NETIRREXIT", "Net IRR Exit{F}TRI net cédé", GetType(Double)),  
—  
New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",  
GetType(Double)), _  
New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _  
New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _  
New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _  
New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _  
New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _  
} _
```

## 6.8.7. CompanyPositions

### 6.8.7.1. COMPANYPOSITIONS

```
New CECubeColumn() { _  
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",  
GetType(DateTime)), _  
New CECubeColumn("CURRENCY", "Currency{F}Devise", GetType(String)), _  
New CECubeColumn("PROPERTY_ID", "Property ID", GetType(String)), _  
New CECubeColumn("COMPANY_ID", "ID Company{F}ID société", GetType(String)), _  
New CECubeColumn("COMPANY_CURR", "Company currency{F}Devise société",  
GetType(String)), _  
New CECubeColumn("COMPANY_NAME", "Company name{F}Nom société",  
GetType(String)), _  
New CECubeColumn("COMPANY_COUNTRY", "Company country{F}Pays société",  
GetType(String)), _
```

New CECubeColumn("COMPANY\_INVESTTYPE", "Company investment type{F}Type investissement société", GetType(String)), \_

New CECubeColumn("COMPANY\_STAGE", "Company stage{F}Devise société", GetType(String)), \_

New CECubeColumn("COMPNBSHARES", "CompNbShares", GetType(Double)), \_

New CECubeColumn("COMPFDSHARES", "CompFDShares", GetType(Double)), \_

New CECubeColumn("UNDRAWNCOMMITMENTINCURR", "UndrawnCommitmentInCurr", GetType(Double)), \_

New CECubeColumn("COMMITMENT", "Commitment", GetType(Double)), \_

New CECubeColumn("UNDRAWNCOMMITMENT", "Remaining Commitment{F}Engagement résiduel", GetType(Double)), \_

New CECubeColumn("INVESTMENT", "Total invested (cash out){F}Total investi (cash out)", GetType(Double)), \_

New CECubeColumn("INVEST", "Net invested{F}Net investi", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTCOST", "Buy fees{F}Frais d'achat", GetType(Double)), \_

New CECubeColumn("ALLOTHERCOST", "Other fees{F}Autres frais", GetType(Double)), \_

New CECubeColumn("EXIT", "Cash in{F}Cash in", GetType(Double)), \_

New CECubeColumn("REIMBURSEMENT", "Proceeds amount{F}Montant cédé", GetType(Double)), \_

New CECubeColumn("SALESPROCEEDS", "Proceeds amount (equity){F}Montant cédé (action)", GetType(Double)), \_

New CECubeColumn("REPAYMENTS", "Proceeds amount (debt){F}Montant cédé (dette)", GetType(Double)), \_

New CECubeColumn("OTHERAMOUNTINCOME", "Sale fees{F}Frais de ventes",  
GetType(Double)), \_

New CECubeColumn("ALLOTHERINCOME", "Total income{F}Total revenus",  
GetType(Double)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Double)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Double)), \_

New CECubeColumn("OTHERINCOME", "Other incomes{F}Autre revenus",  
GetType(Double)), \_

New CECubeColumn("REALIZEDCOST", "Cost of sale{F}Prix de revient cédé",  
GetType(Double)), \_

New CECubeColumn("INTERNALTRANS", "InternalTrans", GetType(Double)), \_

New CECubeColumn("LASTVALUATION", "Last valuation{F}Dernière valorisation",  
GetType(Double)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last valuation date{F}Date de dernière  
valorisation", GetType(DateTime)), \_

New CECubeColumn("ADJVALUATION", "Adjusted valuation{F}Valorisation ajustée",  
GetType(Double)), \_

New CECubeColumn("VALUATIONWOPROV", "ValuationWOProv", GetType(Double)),

—

New CECubeColumn("CURRENTCOST", "Current cost{F}Prix de revient",  
GetType(Double)), \_

New CECubeColumn("CAPITALISEDINTEREST", "CapitalisedInterest",  
GetType(Double)), \_

New CECubeColumn("REALIZEDGAINLOSS", "Realized gain/loss{F}+/- values  
réalisées", GetType(Double)), \_

New CECubeColumn("REALIZEDGLFX", "Realized GL (FX){F}+/- values réalisées (change)", GetType(Double)), \_

New CECubeColumn("REALIZEDGLINV", "Realized GL (market){F}+/- values réalisées (marché)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSS", "Unrealized gain/loss{F}+/- values non réalisées", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSFX", "Unrealized GL (FX){F}+/- values non réalisées (change)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGAINLOSSINV", "Unrealized GL (market){F}+/- values non réalisées (marché)", GetType(Double)), \_

New CECubeColumn("REALIZEDGL", "RealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGL", "UnRealizedGL (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLFX", "UnRealizedGLFX (deprecated)", GetType(Double)), \_

New CECubeColumn("UNREALIZEDGLINV", "UnRealizedGLInv (deprecated)", GetType(Double)), \_

New CECubeColumn("COMMITTEDSHARES", "CommittedShares", GetType(Double)), \_

-

New CECubeColumn("UNDRAWNSHARES", "UndrawnShares", GetType(Double)), \_

New CECubeColumn("SHARESCALLEDINCOMMITMENT", "SharesCalledInCommitment", GetType(Double)), \_

New CECubeColumn("INVESTMENTINCOMMITMENT", "InvestmentInCommitment", GetType(Double)), \_

New CECubeColumn("WRITEOFFCOST", "WriteoffCost", GetType(Double)), \_

New CECubeColumn("PRINCIPAL", "Principal", GetType(Double)), \_

```
New CECubeColumn("BALANCE", "Balance", GetType(Double)), _  
New CECubeColumn("ACCRUALS", "Accruals", GetType(Double)), _  
New CECubeColumn("PIK", "PIK ", GetType(Double)), _  
New CECubeColumn("DUEINTERESTS", "Due interests", GetType(Double)), _  
New CECubeColumn("REALDUEINTERESTS", "Real Due interests", GetType(Double)), _  
—  
New CECubeColumn("PAIDINTERESTS", "Paid interests", GetType(Double)), _  
New CECubeColumn("CASHACCRUALS", "Cash accruals", GetType(Double)), _  
New CECubeColumn("PIKACCRUALS", "PIK accruals", GetType(Double)), _  
New CECubeColumn("CASHACTUALDAYS", "Cash actual days", GetType(Double)), _  
New CECubeColumn("PIKACTUALDAYS", "PIK actual days", GetType(Double)), _  
New CECubeColumn("TOTALVALUE", "Total value{F}Valeur totale", GetType(Double)) _  
} _
```

## 6.8.8. FV\_T\_PORTCO\_TRANSACTIONS

### 6.8.8.1. FV\_T\_PORTCO\_TRANSACTIONS, FV\_T\_CASH\_TRANSACTIONS, FV\_T\_FEES\_TRANSACTIONS

SECURITY\_ID;ID Investment Security{F}ID titre investissement

SECURITY\_COMPANY\_ID;ID Company Security{F}ID titre société

SECURITY\_NAME;Security name{F}Nom du titre

SECURITY\_CURRENCY;Security currency{F}Devise du titre

SECURITY\_TYPE\_ID;ID security type {F}ID type de titre

SECURITY\_TYPE\_CODE;Type

SECURITY\_TYPE\_NAME;Security type {F}Type de titre

SECURITY\_CLASS\_ID;ID Security class{F}ID Classe de titre

SECURITY\_CLASS\_NAME;Security class{F}Classe de titre

SECURITY\_TYPE\_IN\_ND;Security in Non diluted (security type){F}Titre dans le non dilué (Type de titre)

SECURITY\_TYPE\_IN\_FD;Security in Fully diluted (security type){F}Titre dans le tout dilué (Type de titre)

SECURITY\_IN\_ND;Security in Non diluted (security){F}Titre dans le non dilué (Titre)

SECURITY\_USE\_FOR\_ND;Use for ND Calc{F}Utiliser pour calcul du ND

SECURITY\_ROUNDING\_METHOD;Rounding method{F}Règles d'arrondi

SECURITY\_CONVERSION\_MODE;Security conversion mode{F}Mode de conversion du titre

SECURITY\_CONVERSION\_USED\_VALUE;Security conversion used value{F}Valeur utilisée pour conversion du titre

SECURITY\_IPO\_DATE;Security IPO date{F}IPO titre

OPERATION\_ID;ID operation{F}ID opération

OPERATION\_DATE;Operation date{F}Date opération

OPERATION\_TYPE;Operation type{F}Type opération

OPERATION\_TYPE\_NAME;Operation type name{F}Nom du type opération

OPERATION\_DRAFT;Operation draft{F}Opération brouillon

OPERATION\_NAME;Operation name{F}Nom de l'opération

TRANSACTION\_ID;ID transaction{F}ID mouvement

TRANSACTION\_TYPE;Transaction type{F}Type de mouvement

TRANSACTION\_TYPE\_NAME;Transaction type name{F}Nom du type de mouvement

TRANSACTION\_DC;D/C{F}D/C

TRANSACTION\_INTERNAL;Internal{F}Interne

TRANSACTION\_TYPE\_CLASS;Class{F}Classe

TRANSACTION\_CANCELLED;Cancelled

REFERENCE\_DATE;Reference date{F}Date de référence

REFERENCE\_INDEX;Reference index{F}Index de référence

EFFECTIVE\_DATE;Effective date{F}Date effective

IRR\_DATE;IRR date{F}Date TRI

VALUATION;Valuation{F}Valorisation

DRAFT;Draft{F}Brouillon

VALUATION2;Valuation2{F}Valorisation2

COMMITMENT;Commitment{F}Engagement

COMMITMENT2;Commitment2{F}Engagement2

AMOUNT;Amount{F}Montant

AMOUNT2;Amount2{F}Montant2

OTHERAMOUNT;Other amount{F}Autre montant

OTHERAMOUNT2;Other amount2{F}Autre montant2

DUEAMOUNT;Due amount{F}Montant dû

DUEAMOUNT2;Due amount2{F}Montant dû2

COST;Cost{F}Prix de revient

COST2;Cost2{F}Prix de revient2

FORCECOST;Force cost{F}Forcer prix de revient

FORCECOST2;Force cost2{F}Forcer prix de revient2

NB\_SHARES;Nb shares{F}Nombre de titres

FD\_SHARES;Nb shares FD{F}Nombre de titres TD

VT\_SHARES;Voting shares{F}Droits de vote

TRANSACTION\_STAKE;% stake (trans){F}% détention (trans)

TRANSACTION\_FD\_STAKE;% FD stake (trans){F}% détention TD (trans)

INVESTMENT\_ID;ID investment{F}ID investissement

INVESTMENT\_NAME;Investment name{F}Nom investissement

INVESTMENT\_STAKE;% stake (inv){F}% détention(inv)

INVESTMENT\_FD\_STAKE;% FD stake(inv){F}% détention TD(inv)

INVESTOR\_ENTRY\_DATE;Investor entry date{F}Entrée investisseur

INVESTOR\_EXIT\_DATE;Investor exit date{F}Sortie investisseur

INVESTOR\_TYPE;Investor type{F}Type investisseur

INVESTOR\_CATEGORY;Investor category{F}Catégorie investisseur

SECURITY\_TICKER;Security Ticker code{F}Code Ticker du titre

SECURITY\_ISIN;Security ISIN code{F}Code ISIN du titre

SECURITY\_SICOVAM;Security SICOVAM code{F}Code SICOVAM du titre

COMPANY\_ID;IQId

COMPANY\_NAME;Company

COMPANY\_CURRENCY;Currency  
COMPANY\_STAGE;Dealtyp  
COMPANY\_SECTOR0;Sector  
COMPANY\_SUBSECTOR0;Sub sector  
COMPANY\_SECTOR1;Sector  
COMPANY\_SUBSECTOR1;Sub sector  
COMPANY\_SECTOR2;Sector  
COMPANY\_SUBSECTOR2;Sub sector  
COMPANY\_SECTOR3;Sector  
COMPANY\_SUBSECTOR3;Sub sector  
COMPANY\_SECTOR4;Sector  
COMPANY\_SUBSECTOR4;Sub sector  
COMPANY\_TICKER;Ticker  
COMPANY\_ISIN;Sicovam code  
COMPANY\_CREATION\_DATE;Creation date  
COMPANY\_COUNTRY;Country  
COMPANY\_GEOGRAPHY;Region  
FIRST\_INVESTMENT\_STAGE;Stage at investment  
INVESTOR\_FUND\_ID;IQId  
INVESTOR\_FUND\_NAME;Fund  
INVESTOR\_FUND\_GROUPBY\_ID;Group by

INVESTOR\_FUND\_CURRENCY;Currency

INVESTOR\_CONTACT\_ID;IQId

INVESTOR\_CONTACT\_LASTNAME;Last name

INVESTOR\_CONTACT\_FIRSTNAME;First name

INVESTOR\_CONTACT\_CURRENCY;Currency

INVESTOR\_COMPANY\_ID;IQId

INVESTOR\_COMPANY\_NAME;Company

INVESTOR\_COMPANY\_CURRENCY;Currency

INVESTOR\_SUBSCR\_TYPE;

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

VALUATION\_METHOD;Valuation method{F}Méthode de valorisation

LIQUIDITY\_DISCOUNT;%Discount(new/cv)

SECURITY\_CLASS\_NORMALIZED\_ID;Security normalized class ID{F}ID normalisé de classe de titres

COMPANY\_SECTOR;Sector{F}Secteur

COMPANY\_SUBSECTOR;Sub-sector{F}Sous-secteur

COMPANY\_LISTED;Listed?{F}Cotée?

COMPANY\_LISTED\_LABEL;Listed label{F}Libellé cotée

CASH;Cash amount(inv){F}Montant cash(inv)

CASH2;Cash amount(sec){F}Montant cash(sec)

INVESTMENT;Investment amount(inv){F}Montant investi(inv)

INVESTMENT2;Investment amount(sec){F}Montant investi(sec)

INVESTMENT\_WRITEOFF;Investment written off(inv){F}Writeoff investissement(inv)

INVESTMENT\_WRITEOFF2;Investment written off(sec){F}Writeoff investissement(sec)

EXPENSES;Expenses(inv){F}Frais(inv)

EXPENSES2;Expenses(sec){F}Frais(sec)

CAPITALISED\_AMOUNT;Capitalised amount(inv){F}Montant capitalisé(inv)

CAPITALISED\_AMOUNT2;Capitalised amount(sec){F}Montant capitalisé(sec)

PROCEEDS;Proceeds amount(inv){F}Montant cédé(inv)

PROCEEDS2;Proceeds amount(sec){F}Montant cédé(sec)

COSTOFSALE;Cost of sale(inv){F}PR cédé(inv)

COSTOFSALE2;Cost of sale(sec){F}PR cédé(sec)

REALIZED\_GAINLOSS;Realized gain/loss(inv){F}Plus/moins values réalisées(inv)

REALIZED\_GAINLOSS2;Realized gain/loss(sec){F}Plus/moins values réalisées(sec)

INTERESTS;Interests(inv){F}Intérêts(inv)

INTERESTS2;Interests(sec){F}Intérêts(sec)

DIVIDENDS;Dividends(inv){F}Dividendes(inv)

DIVIDENDS2;Dividends(sec){F}Dividendes(sec)

OTHER\_INCOME;Other income(inv){F}Autres revenus(inv)

OTHER\_INCOME2;Other income(sec){F}Autres revenus(sec)

TOTAL\_INCOME;Total income(inv){F}Total revenus(inv)

TOTAL\_INCOME2;Total income(sec){F}Total revenus(sec)

INVESTMENT\_EQUITY;Equity investment(inv){F}Investissement action(inv)

INVESTMENT\_EQUITY2;Equity investment(sec){F}Investissement action(sec)

PROCEEDS\_EQUITY;Equity sale(inv){F}Cession action(inv)

PROCEEDS\_EQUITY2;Equity sale(sec){F}Cession action(sec)

COST\_EQUITY;Equity cost(inv){F}PR action(inv)

COST\_EQUITY2;Equity cost(sec){F}PR action(sec)

NB SHARES\_EQUITY;Equity shares(sec){F}Titres action(sec)

ADJUSTED\_VALUATION\_EQUITY;Equity adjusted valuation(inv){F}Valorisation ajustée action(inv)

ADJUSTED\_VALUATION\_EQUITY2;Equity adjusted valuation(sec){F}Valorisation ajustée action(sec)

INVESTMENT\_BOND;Bond investment(inv){F}Investissement obligation(inv)

INVESTMENT\_BOND2;Bond investment(sec){F}Investissement obligation(sec)

PROCEEDS\_BOND;Bond sale(inv){F}Cession obligation(inv)

PROCEEDS\_BOND2;Bond sale(sec){F}Cession obligation(sec)

COST\_BOND;Bond cost(inv){F}PR obligation(inv)

COST\_BOND2;Bond cost(sec){F}PR obligation(sec)

NB SHARES\_BOND;Bond shares(sec){F}Titres obligation(sec)

ADJUSTED\_VALUATION\_BOND;Bond adjusted valuation(inv){F}Valorisation ajustée obligation(inv)

ADJUSTED\_VALUATION\_BOND2;Bond adjusted valuation(sec){F}Valorisation ajustée obligation(sec)

ADVANCE;Debt advance(inv){F}Avance dette(inv)

ADVANCE2;Debt advance(sec){F}Avance dette(sec)

REIMBURSEMENT;Debt reimbursement(inv){F}Remboursement dette(inv)

REIMBURSEMENT2;Debt reimbursement(sec){F}Remboursement dette(sec)

COST\_LOAN;Loan cost(inv){F}PR dette(inv)

COST\_LOAN2;Loan cost(sec){F}PR dette(sec)

ADJUSTED\_VALUATION\_LOAN;Debt adjusted valuation(inv){F}Valorisation ajustée dette(inv)

ADJUSTED\_VALUATION\_LOAN2;Debt adjusted valuation(sec){F}Valorisation ajustée dette(sec)

INVESTMENT\_OTHER;Other investment(inv){F}Investissement autre(inv)

INVESTMENT\_OTHER2;Other investment(sec){F}Investissement autre(sec)

PROCEEDS\_OTHER;Other sale(inv){F}Cession autre(inv)

PROCEEDS\_OTHER2;Other sale(sec){F}Cession autre(sec)

COST\_OTHER;Other cost(inv){F}PR autre(inv)

COST\_OTHER2;Other cost(sec){F}PR autre(sec)

NB SHARES\_OTHER;Other shares(sec){F}Titres autre(sec)

ADJUSTED\_VALUATION\_OTHER;Other adjusted valuation(inv){F}Valorisation ajustée autre(inv)

ADJUSTED\_VALUATION\_OTHER2;Other adjusted valuation(sec){F}Valorisation ajustée autre(sec)

Loan\_Current\_Balance;Loan Current Balance (Investor currency)

Loan\_Current\_Balance2;Loan Current Balance(Security currency)

Loan\_Advance;Loan advance (Investor currency)

Loan\_Advance2;Loan advance (Security currency)

Loan\_Reimbursement;Loan Reimbursement (Investor currency)

Loan\_Reimbursement2;Loan Reimbursement (Security currency)

Loan\_Capitalized\_Interests;Loan Capitalized Interests (Investor currency)

Loan\_Capitalized\_Interests2;Loan Capitalized Interests (Security currency)

Last\_Coupon\_Payment\_Date;Last coupon payment date

COMPANY\_IPO\_DATE;Security IPO date{F}IPO titre(min)

ID\_CURRTABLE\_FUND\_INVESTOR;

VALUATION\_DATE;Valuation date{F}Date de valorisation

ADJUSTED\_VALUATION;Adjusted valuation(inv){F}Valorisation ajustée(inv)

ADJUSTED\_VALUATION2;Adjusted valuation(sec){F}Valorisation ajustée(sec)

PROVISION;Provision(inv){F}Provision(inv)

PROVISION2;Provision(sec){F}Provision(sec)

UNREALIZED\_GAINLOSS;Unrealized gain/loss(inv){F}+/- value latentes(inv)

UNREALIZED\_GAINLOSS2;Unrealized gain/loss(sec){F}+/- value latentes(sec)

TOTAL\_VALUE;Total value(inv){F}Valeur totale(inv)

TOTAL\_VALUE2;Total value(sec){F}Valeur totale(sec)

TRANSPARENCY;Investor by transparency{F}Investisseur par transparence

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

## 6.8.9. FV\_T\_PORTCO\_POSITIONS\_BY\_SECURITIES

### 6.8.9.1. FV\_T\_PORTCO\_POSITIONS\_BY\_SECURITIES, FV\_T\_CASH\_POSITIONS\_BY\_SECURITIES, FV\_T\_FEES\_POSITIONS\_BY\_SECURITIES

SECURITY\_ID;ID Investment Security{F}ID titre investissement

COMPANY\_ID;IQId

COMPANY\_NAME;Company

COMPANY\_CURRENCY;Currency

COMPANY\_STAGE;Dealtypes

COMPANY\_SECTOR;Sector{F}Secteur

COMPANY\_SUBSECTOR;Sub-sector{F}Sous-secteur

COMPANY\_CREATION\_DATE;Creation date

COMPANY\_COUNTRY;Country

COMPANY\_GEOGRAPHY;Region

COMPANY\_LISTED;Listed?{F}Cotée?

COMPANY\_LISTED\_LABEL;Listed label{F}Libellé cotée

SECURITY\_COMPANY\_ID;ID Company Security{F}ID titre société

SECURITY\_NAME;Security name{F}Nom du titre

SECURITY\_CURRENCY;Security currency{F}Devise du titre

SECURITY\_TICKER;Security Ticker code{F}Code Ticker du titre

SECURITY\_ISIN;Security ISIN code{F}Code ISIN du titre

SECURITY\_SICOVAM;Security SICOVAM code{F}Code SICOVAM du titre

SECURITY\_TYPE\_ID;ID security type {F}ID type de titre

SECURITY\_TYPE\_CODE;Type

SECURITY\_TYPE\_NAME;Security type {F}Type de titre

SECURITY\_CLASS\_ID;ID Security class{F}ID Classe de titre

SECURITY\_CLASS\_NORMALIZED\_ID;Security normalized class ID{F}ID normalisé de classe de titres

SECURITY\_CLASS\_NAME;Security class{F}Classe de titre

SECURITY\_TYPE\_IN\_ND;Security in Non diluted (security type){F}Titre dans le non dilué (Type de titre)

SECURITY\_TYPE\_IN\_FD;Security in Fully diluted (security type){F}Titre dans le tout dilué (Type de titre)

SECURITY\_IN\_ND;Security in Non diluted (security){F}Titre dans le non dilué (Titre)

SECURITY\_USE\_FOR\_ND;Use for ND Calc{F}Utiliser pour calcul du ND

SECURITY\_ROUNDING\_METHOD;Rounding method{F}Règles d'arrondi

SECURITY\_CONVERSION\_MODE;Security conversion mode{F}Mode de conversion du titre

SECURITY\_CONVERSION\_USED\_VALUE;Security conversion used value{F}Valeur utilisée pour conversion du titre

SECURITY\_PRICE\_TYPE;Price type

INVESTMENT\_ID;ID investment{F}ID investissement

INVESTMENT\_NAME;Investment name{F}Nom investissement

FIRST\_INVESTMENT\_STAGE;Stage at investment

INVESTMENT\_STAKE;% stake (inv){F}% détention(inv)

INVESTMENT\_FD\_STAKE;% FD stake(inv){F}% détention TD(inv)

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

INVESTOR\_TYPE;Investor type{F}Type investisseur

INVESTOR\_CATEGORY;Investor category{F}Catégorie investisseur

INVESTOR\_FUND\_ID;IQId

INVESTOR\_FUND\_NAME;Fund

INVESTOR\_FUND\_GROUPBY\_ID;Group by

INVESTOR\_FUND\_CURRENCY;Currency

INVESTOR\_CONTACT\_ID;IQId

INVESTOR\_CONTACT\_LASTNAME;Last name

INVESTOR\_CONTACT\_FIRSTNAME;First name

INVESTOR\_CONTACT\_CURRENCY;Currency

INVESTOR\_COMPANY\_ID;IQId

INVESTOR\_COMPANY\_NAME;Company

INVESTOR\_COMPANY\_CURRENCY;Currency

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

ID\_CURRTABLE\_FUND\_INVESTOR;

ADJUSTED\_VALUATION;Adjusted valuation(inv){F}Valorisation ajustée(inv)

ADJUSTED\_VALUATION2;Adjusted valuation(sec){F}Valorisation ajustée(sec)

UNREALIZED\_GAINLOSS;Unrealized gain/loss(inv){F}+/- values latentes(inv)

UNREALIZED\_GAINLOSS2;Unrealized gain/loss(sec){F}+/- values latentes(sec)

TOTAL\_VALUE;Total value(inv){F}Valeur totale(inv)

TOTAL\_VALUE2;Total value(sec){F}Valeur totale(sec)

PROVISION;Provision(inv){F}Provision(inv)

PROVISION2;Provision(sec){F}Provision(sec)

COMMITMENT;Commitment{F}Engagement

COMMITMENT2;Commitment2{F}Engagement2

NB SHARES;Nb shares{F}Nombre de titres

FD SHARES;Nb shares FD{F}Nombre de titres TD

VT SHARES;Voting shares{F}Droits de vote

CASH;Cash amount(inv){F}Montant cash(inv)

CASH2;Cash amount(sec){F}Montant cash(sec)

COST;Cost(inv){F}Prix de revient(inv)

COST2;Cost(sec){F}Prix de revient(sec)

INVESTMENT;Investment amount(inv){F}Montant investi(inv)

INVESTMENT2;Investment amount(sec){F}Montant investi(sec)

INVESTMENT\_WRITEOFF;Investment written off(inv){F}Writeoff investissement(inv)

INVESTMENT\_WRITEOFF2;Investment written off(sec){F}Writeoff investissement(sec)

EXPENSES;Expenses(inv){F}Frais(inv)

EXPENSES2;Expenses(sec){F}Frais(sec)

CAPITALISED\_AMOUNT;Capitalised amount(inv){F}Montant capitalisé(inv)

CAPITALISED\_AMOUNT2;Capitalised amount(sec){F}Montant capitalisé(sec)

PROCEEDS;Proceeds amount(inv){F}Montant cédé(inv)

PROCEEDS2;Proceeds amount(sec){F}Montant cédé(sec)

COSTOFSALE;Cost of sale(inv){F}PR cédé(inv)

COSTOFSALE2;Cost of sale(sec){F}PR cédé(sec)

REALIZED\_GAINLOSS;Realized gain/loss(inv){F}Plus/moins values réalisées(inv)

REALIZED\_GAINLOSS2;Realized gain/loss(sec){F}Plus/moins values réalisées(sec)

INTERESTS;Interests(inv){F}Intérêts(inv)

INTERESTS2;Interests(sec){F}Intérêts(sec)

DIVIDENDS;Dividends(inv){F}Dividendes(inv)

DIVIDENDS2;Dividends(sec){F}Dividendes(sec)

OTHER\_INCOME;Other income(inv){F}Autres revenus(inv)

OTHER\_INCOME2;Other income(sec){F}Autres revenus(sec)

TOTAL\_INCOME;Total income(inv){F}Total revenus(inv)

TOTAL\_INCOME2;Total income(sec){F}Total revenus(sec)

INVESTMENT\_EQUITY;Equity investment(inv){F}Investissement action(inv)

INVESTMENT\_EQUITY2;Equity investment(sec){F}Investissement action(sec)

PROCEEDS\_EQUITY;Equity sale(inv){F}Cession action(inv)

PROCEEDS\_EQUITY2;Equity sale(sec){F}Cession action(sec)

COST\_EQUITY;Equity cost(inv){F}PR action(inv)

COST\_EQUITY2;Equity cost(sec){F}PR action(sec)

ADJUSTED\_VALUATION\_EQUITY;Equity adjusted valuation(inv){F}Valorisation ajustée action(inv)

ADJUSTED\_VALUATION\_EQUITY2;Equity adjusted valuation(sec){F}Valorisation ajustée action(sec)

NB SHARES\_EQUITY;Equity shares{F}Titres action

INVESTMENT\_BOND;Bond investment(inv){F}Investissement obligation(inv)

INVESTMENT\_BOND2;Bond investment(sec){F}Investissement obligation(sec)

PROCEEDS\_BOND;Bond sale(inv){F}Cession obligation(inv)

PROCEEDS\_BOND2;Bond sale(sec){F}Cession obligation(sec)

COST\_BOND;Bond cost(inv){F}PR obligation(inv)

COST\_BOND2;Bond cost(sec){F}PR obligation(sec)

ADJUSTED\_VALUATION\_BOND;Bond adjusted valuation(inv){F}Valorisation ajustée obligation(inv)

ADJUSTED\_VALUATION\_BOND2;Bond adjusted valuation(sec){F}Valorisation ajustée obligation(sec)

NB SHARES\_BOND;Bond shares{F}Titres obligation

ADVANCE;Debt advance(inv){F}Avance dette(inv)

ADVANCE2;Debt advance(sec){F}Avance dette(sec)

REIMBURSEMENT;Debt reimbursement(inv){F}Remboursement dette(inv)

REIMBURSEMENT2;Debt reimbursement(sec){F}Remboursement dette(sec)

COST\_LOAN;Debt cost(inv){F}PR dette(inv)

COST\_LOAN2;Debt cost(sec){F}PR dette(sec)

ADJUSTED\_VALUATION\_LOAN;Debt adjusted valuation(inv){F}Valorisation ajustée dette(inv)

ADJUSTED\_VALUATION\_LOAN2;Debt adjusted valuation(sec){F}Valorisation ajustée dette(sec)

INVESTMENT\_OTHER;Other investment(inv){F}Investissement autre(inv)

INVESTMENT\_OTHER2;Other investment(sec){F}Investissement autre(sec)

PROCEEDS\_OTHER;Other sale(inv){F}Cession autre(inv)

PROCEEDS\_OTHER2;Other sale(sec){F}Cession autre(sec)

COST\_OTHER;Other cost(inv){F}PR autre(inv)

COST\_OTHER2;Other cost(sec){F}PR autre(sec)

ADJUSTED\_VALUATION\_OTHER;Other adjusted valuation(inv){F}Valorisation ajustée autre(inv)

ADJUSTED\_VALUATION\_OTHER2;Other adjusted valuation(sec){F}Valorisation ajustée autre(sec)

NB SHARES\_OTHER;Other shares({F}Titres autre

Loan\_Advance;Loan advance (Investor currency)

Loan\_Advance2;Loan advance (Security currency)

Loan\_Reimbursement;Loan Reimbursement (Investor currency)

Loan\_Reimbursement2;Loan Reimbursement (Security currency)

Loan\_Capitalized\_Interests;Loan Capitalized Interests (Investor currency)

Loan\_Capitalized\_Interest;Loan Capitalized Interests (Security currency)

Loan\_Current\_Balance;Loan Current Balance (Investor currency)

Loan\_Current\_Balance2;Loan Current Balance(Security currency)

INVESTOR\_ENTRY\_DATE;Investor entry date{F}Entrée investisseur

INVESTOR\_EXIT\_DATE;Investor exit date{F}Sortie investisseur

Last\_Coupon\_Payment\_Date;Last coupon payment date

TRANSACTION\_STAKE;% stake (trans){F}% détention (trans)

TRANSACTION\_FD\_STAKE;% FD stake (trans){F}% détention TD (trans)

VALUATION;Last valuation(inv){F}Dernière valorisation(inv)

VALUATION2;Last valuation(sec){F}Dernière valorisation(sec)

VALUATION\_DATE;Last valuation date{F}Dernière date de valorisation

VALUATION\_METHOD;Last valuation method{F}Dernière méthode de valorisation

LIQUIDITY\_DISCOUNT;Last liquidity discount %{F}Dernier % remise de liquidité

Loan\_Rate;

Loan\_Maturity\_Date;

Loan\_Issue\_Date;

CONVERSION\_DATE;Date{F}Date

CONVERSION\_FROM\_X;X shares{F}X titres

CONVERSION\_TO\_Y1;For Y target shares 1{F}Pour Y titres cible 1

CONVERSION\_TO\_Y2;For Y target shares 2{F}Pour Y titres cible 2

CONVERSION\_TO\_Y3;For Y target shares 3{F}Pour Y titres cible 3

CONVERSION\_TARGET\_SECURITY\_ID1;ID security target 1{F}ID titre cible 1  
CONVERSION\_TARGET\_SECURITY\_ID2;ID security target 2{F}ID titre cible 2  
CONVERSION\_TARGET\_SECURITY\_ID3;ID security target 3{F}ID titre cible 3  
CONVERSION\_USEFORNDFDCALC;ND/FD calculation method{F}Méthode de calcul du ND/TD  
CONVERSION\_ROUNDINGMETHOD;Rounding method{F}Règle d'arrondi  
CONVERSION\_EXCHANGERATE;Explicit exchange rate{F}Taux de change explicite  
CONVERSION\_NOCAPINTEREST;No cap interest{F}Pas d'intérêts capitalisés  
LOANACCRUALS\_PRINCIPAL2;Principal{F}Nominal  
LOANACCRUALS\_WRITEOFF2;Writeoffs{F}Writeoffs  
LOANACCRUALS\_PIK2;PIK{F}Intérêts capitalisés  
LOANACCRUALS\_BALANCE2;Balance{F}Balance  
LOANACCRUALS\_DUEINTEREST2;Due interests{F}Intérêts dûs  
LOANACCRUALS\_PAIDINTEREST2;Paid interests{F}Intérêts payés  
LOANACCRUALS\_CASHACCUAL2;Cash accruals{F}Coupons courus cash  
LOANACCRUALS\_PIKACCUAL2;PIK accruals{F}Coupons courus PIK  
LOANACCRUALS\_CASHDAYS;Cash actual days{F}Nombre de jours courus cash  
LOANACCRUALS\_PIKDAY;PIK actual days{F}Nombre de jours courus PIK  
Loan\_Pool\_Factor;  
Loan\_Pool\_Factor2;  
LOANACCRUALS\_PRINCIPAL;Principal (inv){F}Nominal (inv)

LOANACCRUALS\_WRITEOFF;Writeoffs (inv){F}Writeoffs (inv)

LOANACCRUALS\_PIK;PIK (inv){F}Intérêts capitalisés (inv)

LOANACCRUALS\_BALANCE;Balance (inv){F}Balance (inv)

LOANACCRUALS\_DUEINTEREST;Due interests (inv){F}Intérêts dûs (inv)

LOANACCRUALS\_PAIDINTEREST;Paid interests (inv){F}Intérêts payés (inv)

LOANACCRUALS\_CASHACCRUAL;Cash accruals (inv){F}Coupons courus cash (inv)

LOANACCRUALS\_PIKACCUAL;PIK accruals (inv){F}Coupons courus PIK (inv)

CALC\_NB SHARES;Calc'ed number of shares{F}Nombre de titres calculés

CALC\_FD SHARES;Calc'ed number of shares FD{F}Nombre de titres calculés TD

TRANSPARENCY;Investor by transparency{F}Investisseur par transparence

CALC\_TOTAL\_NB SHARES;Calc'ed total number of shares{F}Total nombre de titres calculés

CALC\_TOTAL\_FD SHARES;Calc'ed total number of shares FD{F}Total nombre de titres calculés TD

CALC\_TOTAL\_NB SHARES\_SECURITY;Calc'ed total number of sharesby security)  
{F}Total nombre de titres calculés (par titre)

CALC\_TOTAL\_FD SHARES\_SECURITY;Calc'ed total number of shares FDby security)  
{F}Total nombre de titres calculés TD (par titre)

CALC\_TOTAL\_COST2\_SECURITY;Calc'ed total cost (by security){F}Total cost (par titre)

ACTIVE;Active status?{F}Statut actif?

ACTIVE\_LABEL;Active status label{F}Libellé statut actif

MULTIPLE;Multiple(inv){F}Multiple(inv)

MULTIPLE2;Multiple(sec){F}Multiple(sec)

IRR;IRR(inv){F}TRI(inv)

IRR2;IRR2(sec){F}TRI(sec)

UNREALIZED\_IRR;Unrealized IRR(inv){F}TRI non cédé(inv)

UNREALIZED\_IRR2;Unrealized IRR2(sec){F}TRI non cédé(sec)

REALIZED\_IRR;Realized IRR(inv){F}TRI cédé(inv)

REALIZED\_IRR2;Realized IRR2(sec){F}TRI cédé(sec)

## 6.8.10. FV\_T\_PORTCO\_POSITIONS\_BY\_INVESTMENT

### 6.8.10.1. FV\_T\_PORTCO\_POSITIONS\_BY\_INVESTMENTS,FV\_T\_CASH\_POSITIONS\_BY\_INVESTMENTS,FV\_T\_FEES\_POSITIONS\_BY\_INVESTMENTS

INVESTMENT\_ID;ID investment{F}ID investissement

COMPANY\_ID;IQId

COMPANY\_NAME;Company

COMPANY\_CURRENCY;Currency

COMPANY\_STAGE;Dealtypes

COMPANY\_SECTOR;Sector{F}Secteur

COMPANY\_SUBSECTOR;Sub-sector{F}Sous-secteur

COMPANY\_CREATION\_DATE;Creation date

COMPANY\_COUNTRY;Country

COMPANY\_GEOGRAPHY;Region

COMPANY\_LISTED;Listed?{F}Cotée?

COMPANY\_LISTED\_LABEL;Listed label{F}Libellé cotée

CALC\_TOTAL\_NB SHARES;Calc'ed total number of shares{F}Total nombre de titres calculés

CALC\_TOTAL\_FD SHARES;Calc'ed total number of shares FD{F}Total nombre de titres calculés TD

INVESTMENT\_NAME;Investment name{F}Nom investissement

FIRST\_INVESTMENT\_STAGE;Stage at investment

INVESTMENT\_STAKE;% stake (inv){F}% détention(inv)

INVESTMENT\_FD\_STAKE;% FD stake(inv){F}% détention TD(inv)

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

INVESTOR\_ENTRY\_DATE;Investor entry date{F}Entrée investisseur

INVESTOR\_EXIT\_DATE;Investor exit date{F}Sortie investisseur

INVESTOR\_TYPE;Investor type{F}Type investisseur

INVESTOR\_CATEGORY;Investor category{F}Catégorie investisseur

INVESTOR\_FUND\_ID;IQId

INVESTOR\_FUND\_NAME;Fund

INVESTOR\_FUND\_GROUPBY\_ID;Group by

INVESTOR\_FUND\_CURRENCY;Currency

INVESTOR\_CONTACT\_ID;IQId

INVESTOR\_CONTACT\_LASTNAME;Last name

INVESTOR\_CONTACT\_FIRSTNAME;First name

INVESTOR\_CONTACT\_CURRENCY;Currency

INVESTOR\_COMPANY\_ID;IQId

INVESTOR\_COMPANY\_NAME;Company

INVESTOR\_COMPANY\_CURRENCY;Currency

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

VALUATION;Last valuation(inv){F}Dernière valorisation(inv)

VALUATION2;Last valuation(sec){F}Dernière valorisation(sec)

CALC\_NB SHARES;Calc'ed number of shares{F}Nombre de titres calculés

CALC\_FD SHARES;Calc'ed number of shares FD{F}Nombre de titres calculés TD

ADJUSTED\_VALUATION;Adjusted valuation(inv){F}Valorisation ajustée(inv)

ADJUSTED\_VALUATION2;Adjusted valuation(corp){F}Valorisation ajustée(corp)

UNREALIZED\_GAINLOSS;Unrealized gain/loss(inv){F}+/- values latentes(inv)

UNREALIZED\_GAINLOSS2;Unrealized gain/loss(corp){F}+/- values latentes(corp)

TOTAL\_VALUE;Total value(inv){F}Valeur totale(inv)

TOTAL\_VALUE2;Total value(corp){F}Valeur totale(corp)

PROVISION;Provision(inv){F}Provision(inv)

PROVISION2;Provision(corp){F}Provision(corp)

COMMITMENT;Commitment{F}Engagement

COMMITMENT2;Commitment2{F}Engagement2

NB SHARES;Nb shares{F}Nombre de titres

FD SHARES;Nb shares FD{F}Nombre de titres TD

VT SHARES;Voting shares{F}Droits de vote

CASH;Cash amount(inv){F}Montant cash(inv)

CASH2;Cash amount(corp){F}Montant cash(corp)

COST;Cost(inv){F}Prix de revient(inv)

COST2;Cost(corp){F}Prix de revient(corp)

INVESTMENT;Investment amount(inv){F}Montant investi(inv)

INVESTMENT2;Investment amount(corp){F}Montant investi(corp)

INVESTMENT\_WRITEOFF;Investment written off(inv){F}Writeoff investissement(inv)

INVESTMENT\_WRITEOFF2;Investment written off(corp){F}Writeoff investissement(corp)

EXPENSES;Expenses(inv){F}Frais(inv)

EXPENSES2;Expenses(corp){F}Frais(corp)

CAPITALISED\_AMOUNT;Capitalised amount(inv){F}Montant capitalisé(inv)

CAPITALISED\_AMOUNT2;Capitalised amount(corp){F}Montant capitalisé(corp)

PROCEEDS;Proceeds amount(inv){F}Montant cédé(inv)

PROCEEDS2;Proceeds amount(corp){F}Montant cédé(corp)

COSTOFSALE;Cost of sale(inv){F}PR cédé(inv)

COSTOFSALE2;Cost of sale(corp){F}PR cédé(corp)

REALIZED\_GAINLOSS;Realized gain/loss(inv){F}Plus/moins values réalisées(inv)

REALIZED\_GAINLOSS2;Realized gain/loss(corp){F}Plus/moins values réalisées(corp)

INTERESTS;Interests(inv){F}Intérêts(inv)

INTERESTS2;Interests(corp){F}Intérêts(corp)

DIVIDENDS;Dividends(inv){F}Dividendes(inv)

DIVIDENDS2;Dividends(corp){F}Dividendes(corp)

OTHER\_INCOME;Other income(inv){F}Autres revenus(inv)

OTHER\_INCOME2;Other income(corp){F}Autres revenus(corp)

TOTAL\_INCOME;Total income(inv){F}Total revenus(inv)

TOTAL\_INCOME2;Total income(corp){F}Total revenus(corp)

INVESTMENT\_EQUITY;Equity investment(inv){F}Investissement action(inv)

INVESTMENT\_EQUITY2;Equity investment(corp){F}Investissement action(corp)

PROCEEDS\_EQUITY;Equity sale(inv){F}Cession action(inv)

PROCEEDS\_EQUITY2;Equity sale(corp){F}Cession action(corp)

COST\_EQUITY;Equity cost(inv){F}PR action(inv)

COST\_EQUITY2;Equity cost(corp){F}PR action(corp)

ADJUSTED\_VALUATION\_EQUITY;Equity adjusted valuation(inv){F}Valorisation ajustée action(inv)

ADJUSTED\_VALUATION\_EQUITY2;Equity adjusted valuation(corp){F}Valorisation ajustée action(corp)

NB SHARES\_EQUITY;Equity shares{F}Titres action

INVESTMENT\_BOND;Bond investment(inv){F}Investissement obligation(inv)

INVESTMENT\_BOND2;Bond investment(corp){F}Investissement obligation(corp)

PROCEEDS\_BOND;Bond sale(inv){F}Cession obligation(inv)

PROCEEDS\_BOND2;Bond sale(corp){F}Cession obligation(corp)

COST\_BOND;Bond cost(inv){F}PR obligation(inv)

COST\_BOND2;Bond cost(corp){F}PR obligation(corp)

ADJUSTED\_VALUATION\_BOND;Bond adjusted valuation(inv){F}Valorisation ajustée obligation(inv)

ADJUSTED\_VALUATION\_BOND2;Bond adjusted valuation(corp){F}Valorisation ajustée obligation(corp)

NB SHARES\_BOND;Bond shares({F}Titres obligation

ADVANCE;Debt advance(inv){F}Avance dette(inv)

ADVANCE2;Debt advance(corp){F}Avance dette(corp)

REIMBURSEMENT;Debt reimbursement(inv){F}Remboursement dette(inv)

REIMBURSEMENT2;Debt reimbursement(corp){F}Remboursement dette(corp)

COST\_LOAN;Debt cost(inv){F}PR dette(inv)

COST\_LOAN2;Debt cost(corp){F}PR dette(corp)

ADJUSTED\_VALUATION\_LOAN;Debt adjusted valuation(inv){F}Valorisation ajustée dette(inv)

ADJUSTED\_VALUATION\_LOAN2;Debt adjusted valuation(corp){F}Valorisation ajustée dette(corp)

INVESTMENT\_OTHER;Other investment(inv){F}Investissement autre(inv)

INVESTMENT\_OTHER2;Other investment(corp){F}Investissement autre(corp)

PROCEEDS\_OTHER;Other sale(inv){F}Cession autre(inv)

PROCEEDS\_OTHER2;Other sale(corp){F}Cession autre(corp)

COST\_OTHER;Other cost(inv){F}PR autre(inv)

COST\_OTHER2;Other cost(corp){F}PR autre(corp)

ADJUSTED\_VALUATION\_OTHER;Other adjusted valuation(inv){F}Valorisation ajustée autre(inv)

ADJUSTED\_VALUATION\_OTHER2;Other adjusted valuation(corp){F}Valorisation ajustée autre(corp)

NB SHARES\_OTHER;Other shares({F}Titres autre

Loan\_Advance;Loan advance (Investor currency)

Loan\_Advance2;Loan advance (Investee currency)

Loan\_Reimbursement;Loan Reimbursement (Investor currency)

Loan\_Reimbursement2;Loan Reimbursement (Investee currency)

Loan\_Capitalized\_Interests;Loan Capitalized Interests (Investor currency)

Loan\_Capitalized\_Interests2;Loan Capitalized Interests (Investee currency)

Loan\_Current\_Balance;Loan Current Balance (Investor currency)

Loan\_Current\_Balance2;Loan Current Balance(Investee currency)

LOANACCRUALS\_PRINCIPAL;Principal (inv){F}Nominal (inv)

LOANACCRUALS\_PRINCIPAL2;Principal (corp){F}Nominal (corp)

LOANACCRUALS\_WRITEOFF;Writeoffs (inv){F}Writeoffs (inv)

LOANACCRUALS\_WRITEOFF2;Writeoffs (corp){F}Writeoffs (corp)

LOANACCRUALS\_PIK;PIK (inv){F}Intérêts capitalisés (inv)

LOANACCRUALS\_PIK2;PIK (corp){F}Intérêts capitalisés (corp)

LOANACCRUALS\_BALANCE;Balance (inv){F}Balance (inv)

LOANACCRUALS\_BALANCE2;Balance (corp){F}Balance (corp)

LOANACCRUALS\_DUEINTEREST;Due interests (inv){F}Intérêts dûs (inv)

LOANACCRUALS\_DUEINTEREST2;Due interests (corp){F}Intérêts dûs (corp)

LOANACCRUALS\_PAIDINTEREST;Paid interests (inv){F}Intérêts payés (inv)

LOANACCRUALS\_PAIDINTEREST2;Paid interests (corp){F}Intérêts payés (corp)

LOANACCRUALS\_CASHACCRUAL;Cash accruals (inv){F}Coupons courus cash (inv)

LOANACCRUALS\_CASHACCRUAL2;Cash accruals (corp){F}Coupons courus cash (corp)

LOANACCRUALS\_PIKACCRUAL;PIK accruals (inv){F}Coupons courus PIK (inv)

LOANACCRUALS\_PIKACCRUAL2;PIK accruals (corp){F}Coupons courus PIK (corp)

VALUATION\_DATE;Last valuation date{F}Dernière date de valorisation

TRANSACTION\_STAKE;% stake (trans){F}% détention (trans)

TRANSACTION\_FD\_STAKE;% FD stake (trans){F}% détention TD (trans)

VALUATION\_METHOD\_INV;Last valuation method{F}Dernière méthode de valorisation(last)

CALC\_STAKE;% stake{F}% détention

CALC\_FD\_STAKE;% FD stake{F}% détention TD

ACTIVE;Active status code{F}Code statut actif

ACTIVE\_LABEL;Active status{F}Statut actif

MULTIPLE;Multiple(inv){F}Multiple(inv)

TRANSPARENCY;Investor by transparency{F}Investisseur par transparence

IRR;IRR(inv){F}TRI(inv)

REALIZED\_IRR;Realized - Unrealized IRR{F}TRI Cédé - Non Cédé

UNREALIZEDIRR;Realized - Unrealized IRR{F}TRI Cédé - Non Cédé

MULTIPLE2;Multiple(sec){F}Multiple(sec)

IRR2;IRR2(sec){F}TRI(sec)

REALIZEDIRR2;Realized IRR2(sec){F}TRI cédé(sec)

UNREALIZEDIRR2;Unrealized IRR2(sec){F}TRI non cédé(sec)

## 6.8.11. FV\_T\_PORTCO\_POSITIONS\_BY\_INVESTORS

### 6.8.11.1. FV\_T\_PORTCO\_POSITIONS\_BY\_INVESTORS,FV\_T\_CASH\_POSITIONS\_BY\_INVESTORS,FV\_T\_FEES\_POSITIONS\_BY\_INVESTORS

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

INVESTOR\_ENTRY\_DATE;Investor entry date{F}Entrée investisseur

INVESTOR\_EXIT\_DATE;Investor exit date{F}Sortie investisseur

INVESTOR\_TYPE;Investor type{F}Type investisseur

INVESTOR\_CATEGORY;Investor category{F}Catégorie investisseur

INVESTOR\_FUND\_ID;IQId

INVESTOR\_FUND\_GROUPBY\_ID;Group by

INVESTOR\_FUND\_NAME;Fund

INVESTOR\_FUND\_CURRENCY;Currency

INVESTOR\_CONTACT\_ID;IQId

INVESTOR\_CONTACT\_LASTNAME;Last name

INVESTOR\_CONTACT\_FIRSTNAME;First name

INVESTOR\_CONTACT\_CURRENCY;Currency

INVESTOR\_COMPANY\_ID;IQId

INVESTOR\_COMPANY\_NAME;Company

INVESTOR\_COMPANY\_CURRENCY;Currency

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

ADJUSTED\_VALUATION;Adjusted valuation(inv){F}Valorisation ajustée(inv)

UNREALIZED\_GAINLOSS;Unrealized gain/loss(inv){F}+/- values latentes(inv)

TOTAL\_VALUE;Total value(inv){F}Valeur totale(inv)

PROVISION;Provision(inv){F}Provision(inv)

COMMITMENT;Commitment{F}Engagement

NB SHARES;Nb shares{F}Nombre de titres

FD SHARES;Nb shares FD{F}Nombre de titres TD

VT SHARES;Voting shares{F}Droits de vote

CASH;Cash amount(inv){F}Montant cash(inv)

COST;Cost(inv){F}Prix de revient(inv)

INVESTMENT;Investment amount(inv){F}Montant investi(inv)

INVESTMENT\_WRITEOFF;Investment written off(inv){F}Writeoff investissement(inv)

EXPENSES;Expenses(inv){F}Frais(inv)

CAPITALISED\_AMOUNT;Capitalised amount(inv){F}Montant capitalisé(inv)

PROCEEDS;Proceeds amount(inv){F}Montant cédé(inv)

COSTOFSALE;Cost of sale(inv){F}PR cédé(inv)

REALIZED\_GAINLOSS;Realized gain/loss(inv){F}Plus/moins values réalisées(inv)

INTERESTS;Interests(inv){F}Intérêts(inv)

DIVIDENDS;Dividends(inv){F}Dividendes(inv)

OTHER\_INCOME;Other income(inv){F}Autres revenus(inv)

TOTAL\_INCOME;Total income(inv){F}Total revenus(inv)

INVESTMENT\_EQUITY;Equity investment(inv){F}Investissement action(inv)

PROCEEDS\_EQUITY;Equity sale(inv){F}Cession action(inv)

COST\_EQUITY;Equity cost(inv){F}PR action(inv)

ADJUSTED\_VALUATION\_EQUITY;Equity adjusted valuation(inv){F}Valorisation ajustée action(inv)

NB SHARES\_EQUITY;Equity shares{F}Titres action

INVESTMENT\_BOND;Bond investment(inv){F}Investissement obligation(inv)

PROCEEDS\_BOND;Bond sale(inv){F}Cession obligation(inv)

COST\_BOND;Bond cost(inv){F}PR obligation(inv)

ADJUSTED\_VALUATION\_BOND;Bond adjusted valuation(inv){F}Valorisation ajustée obligation(inv)

NB SHARES\_BOND;Bond shares{F}Titres obligation

ADVANCE;Debt advance(inv){F}Avance dette(inv)

REIMBURSEMENT;Debt reimbursement(inv){F}Remboursement dette(inv)

COST\_LOAN;Debt cost(inv){F}PR dette(inv)

ADJUSTED\_VALUATION\_LOAN;Debt adjusted valuation(inv){F}Valorisation ajustée dette(inv)

INVESTMENT\_OTHER;Other investment(inv){F}Investissement autre(inv)

PROCEEDS\_OTHER;Other sale(inv){F}Cession autre(inv)

COST\_OTHER;Other cost(inv){F}PR autre(inv)

ADJUSTED\_VALUATION\_OTHER;Other adjusted valuation(inv){F}Valorisation ajustée autre(inv)

NB SHARES\_OTHER;Other shares({F}Titres autre

Loan\_Advance;Loan advance (Investor currency)

Loan\_Reimbursement;Loan Reimbursement (Investor currency)

Loan\_Capitalized\_Interests;Loan Capitalized Interests (Investor currency)

Loan\_Current\_Balance;Loan Current Balance (Investor currency)

LOANACCRUALS\_PRINCIPAL;Principal (inv){F}Nominal (inv)

LOANACCRUALS\_WRITEOFF;Writeoffs (inv){F}Writeoffs (inv)

LOANACCRUALS\_PIK;PIK (inv){F}Intérêts capitalisés (inv)

LOANACCRUALS\_BALANCE;Balance (inv){F}Balance (inv)

LOANACCRUALS\_DUEINTEREST;Due interests (inv){F}Intérêts dûs (inv)

LOANACCRUALS\_PAIDINTEREST;Paid interests (inv){F}Intérêts payés (inv)

LOANACCRUALS\_CASHACCRUAL;Cash accruals (inv){F}Coupons courus cash (inv)

LOANACCRUALS\_PIKACCRUAL;PIK accruals (inv){F}Coupons courus PIK (inv)

MULTIPLE;Multiple{F}Multiple

TRANSPARENCY;Investor by transparency{F}Investisseur par transparence

IRR;IRR(inv){F}TRI(inv)

PORTFOLIO\_ACTIVE\_IRR;Active IRR(inv){F}TRI(inv) courant

PORTFOLIO\_TOTAL\_PARTIAL\_EXIT\_IRR;Total and Partial Exit IRR(inv){F}TRI(inv)  
cessions partielles et totales

PORTFOLIO\_REALIZED\_IRR;Realized IRR(inv){F}TRI(inv) réalisé

PORTFOLIO\_PARTIAL\_EXIT\_IRR;Partial exit IRR(inv){F}TRI(inv) cession partielle

PORTFOLIO\_UNREALIZED\_IRR;Unrealized IRR(inv){F}TRI(inv) non réalisé

REALIZED\_IRR;Realized - Unrealized IRR{F}TRI Cédé - Non Cédé

UNREALIZED\_IRR;Realized - Unrealized IRR{F}TRI Cédé - Non Cédé

## 6.9. Data tables produced by the Funds CE (FCE)

The eFront Invest calculation engine for funds produces the following tables:

- FundCashFlows
- FundOperations
- Section 6.9.2, “FundShareOperations”
- Section 6.9.3, “FundInvestorOperations”
- FundInvestorShareOperations
- FundPositions and FundPositionsWithoutCarried
- FundInvestorPositions
- FundSharePositions
- FundInvestorSharePositions
- ► InvestorPositions

## INVESTORPOSITIONS

```
New CECubeColumn() {_
    New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",
        GetType(DateTime)), _
    New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",
        GetType(String)), _
    New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",
        GetType(String)), _
    New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",
        GetType(String)), _
    New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de
        l'investisseur", GetType(String)), _
    New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des
        montants", GetType(String)), _
    New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",
        GetType(Decimal)), _
    New CECubeColumn("REMAININGCOMMITMENT", "Remaining
        commitment{F}Engagement résiduel", GetType(Decimal)), _
    New CECubeColumn("PERCENTCALLED", "% called{F}% appelé",
        GetType(Double)), _}
```

```
New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), _
New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), _
New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), _
New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé",
    GetType(Decimal)), _
New CECubeColumn("CAPITALCALLED_INSIDE", "Capital called inside
commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), _
New CECubeColumn("CAPITALCALLED_OUTSIDE", "Capital called outside
commitment{F}Capital appelé hors engagement", GetType(Decimal)), _
New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi",
    GetType(Decimal)), _
New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de
gestion", GetType(Decimal)), _
New CECubeColumn("MANAGEMENTFEES_INSIDE", "Management fees inside
commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), _
New CECubeColumn("MANAGEMENTFEES_OUTSIDE", "Management fees outside
commitment{F}Frais de gestion hors engagement", GetType(Decimal)), _
New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)),

—
New CECubeColumn("OTHERFEES_INSIDE", "Other fees inside
commitment{F}Autres frais dans l'engagement", GetType(Decimal)), _
New CECubeColumn("OTHERFEES_OUTSIDE", "Other fees outside
commitment{F}Autres frais hors engagement", GetType(Decimal)), _
New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions",
    GetType(Decimal)), _
New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital",
    GetType(Decimal)), _
New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital
rappelable", GetType(Decimal)), _
New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",
    GetType(Decimal)), _
New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
    GetType(Decimal)), _
```

```

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date
NAV", GetType(DateTime)), _
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",
GetType(Decimal)), _
New CECubeColumn("IRRNET", "Net IRR{F}TRI net", GetType(Double)), _
New CECubeColumn("IRRNETREALIZED", "Realized net IRR{F}TRI net cédé",
GetType(Double)), _
New CECubeColumn("IRRNETUNREALIZED", "Unrealized net IRR{F}TRI net non
cédé", GetType(Double)), _
New CECubeColumn("TVPI", "TVPI", GetType(Double)), _
New CECubeColumn("DPI", "DPI", GetType(Double)), _
New CECubeColumn("RVPI", "RVPI", GetType(Double)), _
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)) _
}_

```

- ▶ [MasterfundInvestorPositions](#)

## MASTERFUNDINVESTORPOSITIONS

```

New CECubeColumn() {_
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",
GetType(DateTime)), _
New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _
New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds",
GetType(String)), _
New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",
GetType(String)), _
New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",
GetType(String)), _
New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",
GetType(String)), _
New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",
GetType(String)), _
New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de
l'investisseur", GetType(String)), _

```

```
New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des  
montants", GetType(String)), _  
New CECubeColumn("INVESTMENTEXITDATE", "Exit date{F}Date de sortie",  
GetType(Date)), _  
New CECubeColumn("ISEXITED", "Exit?{F}Sorti ?", GetType(Boolean)), _  
New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",  
GetType(Decimal)), _  
New CECubeColumn("REMAININGCOMMITMENT", "Remaining  
commitment{F}Engagement résiduel", GetType(Decimal)), _  
New CECubeColumn("PERCENTCALLED", "% called{F}% appelé",  
GetType(Double)), _  
New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), _  
New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), _  
New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), _  
New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé",  
GetType(Decimal)), _  
New CECubeColumn("CAPITALCALLED_INSIDE", "Capital called inside  
commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), _  
New CECubeColumn("CAPITALCALLED_OUTSIDE", "Capital called outside  
commitment{F}Capital appelé hors engagement", GetType(Decimal)), _  
New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi",  
GetType(Decimal)), _  
New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de  
gestion", GetType(Decimal)), _  
New CECubeColumn("MANAGEMENTFEES_INSIDE", "Management fees inside  
commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), _  
New CECubeColumn("MANAGEMENTFEES_OUTSIDE", "Management fees outside  
commitment{F}Frais de gestion hors engagement", GetType(Decimal)), _  
New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), _  
  
New CECubeColumn("OTHERFEES_INSIDE", "Other fees inside  
commitment{F}Autres frais dans l'engagement", GetType(Decimal)), _  
New CECubeColumn("OTHERFEES_OUTSIDE", "Other fees outside  
commitment{F}Autres frais hors engagement", GetType(Decimal)), _
```

```

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions",
GetType(Decimal)), _
New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital",
GetType(Decimal)), _
New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital
rappelable", GetType(Decimal)), _
New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",
GetType(Decimal)), _
New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
GetType(Decimal)), _
New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date
NAV", GetType(DateTime)), _
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",
GetType(Decimal)), _
New CECubeColumn("IRRNET", "Net IRR{F}TRI net", GetType(Double)), _
New CECubeColumn("TVPI", "TVPI", GetType(Double)), _
New CECubeColumn("DPI", "DPI", GetType(Double)), _
New CECubeColumn("RVPI", "RVPI", GetType(Double)), _
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)), _
New CECubeColumn("STAKE", "Stake %{F}% détention", GetType(Double)) _
}_

```

The calculation engine produces also the same tables as the datamart. The calculation engine generates these tables in a more efficient way. These tables are:

- ▶ [FV\\_T\\_FUND\\_TRANSACTIONS](#)

## FV\_T\_FUND\_TRANSACTIONS

OPERATION\_ID;ID fund operation{F}ID opération fonds  
OPERATION\_NAME;Name fund operation{F}Libellé opération fonds  
FUND\_ID;ID fund{F}ID Fonds

FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
REFERENCE\_DATE;Date{F}Date  
INDEX;Index{F}Index  
TYPE;Type{F}Type  
OPERATION\_TYPE;Formatted type{F}Type formatté  
TRANSACTION\_CANCELLED;Cancelled  
DRAFT;Draft{F}Brouillon  
IRR\_DATE;Date for IRR  
EXPL\_STAKE;Explicit stake{F}% détention explicite  
EXPL\_STAKE\_NC;Explicit stake no carried{F}% détention explicite hors carried  
VALUATION;Valuation{F}Valorisation  
VALUATION2;Valuation(curr2)  
COST;Cost{F}Prix de revient  
COST2;Cost(curr2)  
INVESTMENT\_ID;ID investment{F}ID Investissement  
COMMITMENT;Commitment{F}Engagement  
COMMITMENT1;Commitment(1){F}Engagement(1)  
COMMITMENT2;Commitment(2){F}Engagement(2)  
REMAINING\_COMMITMENT;Remaining commitment  
REMAINING\_COMMITMENT1;Remaining commitment(1)  
REMAINING\_COMMITMENT2;Remaining commitment(2)  
CASH;Cash amount{F}Montant cash  
CASH1;Cash amount(1){F}Montant cash(1)  
CASH2;Cash amount(2){F}Montant cash(2)  
CAPITAL\_CALL;Capital call{F}Capital appelé  
CAPITAL\_CALL1;Capital call(1){F}Capital appelé(1)  
CAPITAL\_CALL2;Capital call(2){F}Capital appelé(2)  
CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Capital appelé hors engagement  
CAPITAL\_CALL\_OUTSIDE\_COMM1;Capital call outside commitment(1){F}Capital appelé hors engagement(1)  
CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(2){F}Capital appelé hors engagement(2)  
DISTRIBUTION;Distribution{F}Distribution  
DISTRIBUTION1;Distribution(1){F}Distribution(1)

DISTRIBUTION2;Distribution(2){F}Distribution(2)  
MANAGEMENT\_FEES;Management fees{F}Frais de gestion  
MANAGEMENT\_FEES1;Management fees(1){F}Frais de gestion(1)  
MANAGEMENT\_FEES2;Management fees(2){F}Frais de gestion(2)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement  
MANAGEMENT\_FEES\_OUTSIDE\_COMM1;Management fees outside commitment(1){F}Frais de gestion hors engagement(1)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(2){F}Frais de gestion hors engagement(2)  
OTHER\_FEES;Other fees{F}Autres frais  
OTHER\_FEES1;Other fees(1){F}Autres frais(1)  
OTHER\_FEES2;Other fees(2){F}Autres frais(2)  
OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement  
OTHER\_FEES\_OUTSIDE\_COMM1;Other fees outside commitment(1){F}Autres frais hors engagement(1)  
OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(2){F}Autres frais hors engagement(2)  
RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé  
RETURN\_OF\_CAPITAL1;Return of capital(1){F}Capital remboursé(1)  
RETURN\_OF\_CAPITAL2;Return of capital(2){F}Capital remboursé(2)  
REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire  
REDRAWABLE\_CAPITAL1;Redrawable call(1){F}Distribution temporaire(1)  
REDRAWABLE\_CAPITAL2;Redrawable call(2){F}Distribution temporaire(2)  
CARRY;Carry{F}Intérêt prioritaire  
CARRY1;Carry(1){F}Intérêt prioritaire(1)  
CARRY2;Carry(2){F}Intérêt prioritaire(2)  
GAIN\_LOSS;Gain/Loss{F}Plus/moins value  
GAIN\_LOSS1;Gain/Loss(1){F}Plus/moins value(1)  
GAIN\_LOSS2;Gain/Loss(2){F}Plus/moins value(2)  
INTERESTS;Interests{F}Intérêts  
INTERESTS1;Interests(1){F}Intérêts(1)  
INTERESTS2;Interests(2){F}Intérêts(2)  
DIVIDENDS;Dividends{F}Dividendes

DIVIDENDS1;Dividends(1){F}Dividendes(1)  
DIVIDENDS2;Dividends(2){F}Dividendes(2)  
OTHER\_INCOME;Other income{F}Autres revenus  
OTHER\_INCOME1;Other income(1){F}Autres revenus(1)  
OTHER\_INCOME2;Other income(2){F}Autres revenus(2)  
TRANSACTION\_COSTS;Transaction costs{F}Coûts de transaction  
TRANSACTION\_COSTS1;Transaction costs(1){F}Coûts de transaction(1)  
TRANSACTION\_COSTS2;Transaction costs(2){F}Coûts de transaction(2)  
COMPANY\_ID;Company ID{F}ID société  
COMPANY\_NAME;Company name{F}Société  
DIK SHARES;Number of shares{F}Nombre de titres  
DIK\_VALUE\_PER\_SHARE;Value per share{F}Valeur du titre  
DIK\_COST\_PER\_SHARE;Cost per share{F}Prix de revient du titre  
SOS\_CASH;SOS Cash amount{F}Montant cession titres  
SOS\_CASH1;SOS Cash amount(1){F}Montant cession titres(1)  
SOS\_CASH2;SOS Cash amount(2){F}Montant cession titres(2)  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement  
GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intérêts hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif  
UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée  
TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
SHAREINDEX;  
VALUATION1;  
COST1;  
SECURITY\_CURRENCY;Share currency{F}Devise part  
FUND\_NAME;Fund name{F}Nom du fonds

FUND\_CURRENCY;Fund currency{F}Devise fonds  
INIT\_COMMITMENT;Initial commitment{F}Engagement initial  
INVESTMENT\_NAME;Investment name{F}Nom investissement  
INVESTMENT\_TYPE\_CODE;Type  
INVESTMENT\_TYPE;Type  
INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement  
INVESTOR\_FUND\_ID;ID fund investor{F}ID fonds Investisseur  
INVESTOR\_FUND\_NAME;Name fund investor{F}Nom fonds Investisseur  
INVESTOR\_FUND\_CURRENCY;Currency fund investor{F}Devise fonds Investisseur  
INVESTOR\_FUND\_STATUS;Status fund investor{F}Statut fonds Investisseur  
INVESTOR\_COMPANY\_ID;ID company investor{F}ID société Investisseur  
INVESTOR\_COMPANY\_NAME;Name company investor{F}Nom société Investisseur  
INVESTOR\_COMPANY\_CURRENCY;Currency company investor{F}Devise société Investisseur  
INVESTOR\_COMPANY\_STATUS;Status company investor{F}Statut société Investisseur  
INVESTOR\_CONTACT\_ID;ID contact investor{F}ID contact Investisseur  
INVESTOR\_CONTACT\_LASTNAME;Lastname contact investor{F}Nom contact Investisseur  
INVESTOR\_CONTACT\_FIRSTNAME;Firstname contact investor{F}Prénom contact Investisseur  
INVESTOR\_CONTACT\_CURRENCY;Currency contact investor{F}Devise contact Investisseur  
INVESTOR\_CONTACT\_STATUS;Status contact investor{F}Statut contact Investisseur  
ID\_CURRTABLE\_FUND;  
INVESTOR\_ID;ID investor{F}ID investisseur  
INVESTOR\_NAME;Investor name{F}Nom investisseur  
INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur  
RATE;FX rate (Investor / Share){F}Taux de change (Investisseur / Part)  
RATE1;FX rate (Fund / Share){F}Taux de change (Fonds / Part)  
CASH\_OUT;Cash out  
CASH\_OUT1;Cash out(1)  
CASH\_OUT2;Cash out(2)  
CASH\_IN;Cash in  
CASH\_IN1;Cash in(1)

CASH\_IN2;Cash in(2)  
ADJUSTED\_VALUATION;Adjusted valuation{F}Valorisation ajustée  
ADJUSTED\_VALUATION1;Adjusted valuation(1){F}Valorisation ajustée(1)  
ADJUSTED\_VALUATION2;Adjusted valuation(2){F}Valorisation ajustée(2)  
LAST\_VALUATION\_DATE;Last valuation date{F}Date dernière valorisation  
SECURITY\_COMMITMENT;Nb shares committed{F}Nb parts engagées  
SECURITY\_NOMINAL;Nominal ajdusment{F}Ajustement nominal  
SECURITY\_NOMINAL1;Nominal ajdusment(1){F}Ajustement nominal(1)  
SECURITY\_NOMINAL2;Nominal ajdusment(2){F}Ajustement nominal(2)  
SECURITY\_NOMINAL\_CALLED;Nominal called{F}Nominal appelé  
SECURITY\_NOMINAL\_CALLED1;Nominal called(1){F}Nominal appelé(1)  
SECURITY\_NOMINAL\_CALLED2;Nominal called(2){F}Nominal appelé(2)  
SECURITY\_NOMINAL\_BOUGHTBACK;Nominal bought back{F}Nominal racheté  
SECURITY\_NOMINAL\_BOUGHTBACK1;Nominal bought back(1){F}Nominal racheté(1)  
SECURITY\_NOMINAL\_BOUGHTBACK2;Nominal bought back(2){F}Nominal racheté(2)  
SECURITY\_VALUE;Security value{F}Valeur part  
SECURITY\_VALUE1;Security value(1){F}Valeur part(1)  
SECURITY\_VALUE2;Security value(2){F}Valeur part(2)  
SECURITY\_CALLED;Security called{F}Appelé part  
SECURITY\_CALLED1;Security called(1){F}Appelé part(1)  
SECURITY\_CALLED2;Security called(2){F}Appelé part(2)  
SECURITY\_DISTRIBUTED;Security distributed{F}Distribué part  
SECURITY\_DISTRIBUTED1;Security distributed(1){F}Distribué part(1)  
SECURITY\_DISTRIBUTED2;Security distributed(2){F}Distribué part(2)  
NB SHARES;Nb shares{F}Nb parts  
NB SHARES\_CALLED;Nb shares called{F}Nb parts appelées  
NB SHARES\_BOUGHTBACK;Nb shares bought back{F}Nb parts rachetées  
SHARE\_ID;Share ID{F}ID Part  
SHARE\_TICKER;Share Ticker code{F}Code Ticker de la part  
SHARE\_ISIN;Share ISIN code{F}Code ISIN de la part  
SHARE\_SICOVAM;Share SICOVAM code{F}Code SICOVAM de la part  
USE\_SECONDARY\_CURRENCY;Subscription currency is commitment currency

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Capital appelé dans l'engagement  
CAPITAL\_CALL\_INSIDE\_COMM1;Capital call inside commitment(1){F}Capital appelé dans l'engagement(1)  
CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Capital appelé dans l'engagement(2)  
SECURITY\_FUND\_ID;ID Fund share{F}ID Part  
SECURITY\_ISCARRIED;Carried share{F}Part de carried  
SECURITY\_NAME;Shortname{F}Abbréviation  
SECURITY\_COMPLETE\_NAME;Name{F}Nom  
SECURITY\_PRICE1;Price{F}Nominal  
NBSHARE\_ROUNDING;Number of shares rounding{F}Arrondi du nombre de parts  
NBSHARE\_ROUNDING\_TYPE;Number of share rounding mode  
SHAREPRICE\_ROUNDING;Share price rounding{F}Arrondi du nominal  
SHAREPRICE\_ROUNDING\_TYPE;Share price rounding mode{F}Type d'arrondi par part  
IS\_ROUNDINGSHARE\_CLASS;Is rounding share class{F}Classe réceptionnant les arrondis  
COMMITMENT\_IN\_AMOUNT;Commitment in amount  
SECURITY\_ID;Security ID{F}ID part  
SECURITY\_PRICE;Price{F}Nominal  
SECURITY\_PRICE2;Price(2){F}Nominal(2)  
NB SHARES\_ROUNDED;Nb shares rounded{F}Nb parts arrondi  
NB SHARES\_CALLED\_ROUNDED;Nb shares called rounded{F}Nb parts appelées arrondi  
NB SHARES\_BOUGHTBACK\_ROUNDED;Nb shares bought back rounded{F}Nb parts rachetées arrondi  
YEAR;Year{F}Année  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
• ► [FV\\_T\\_FUND\\_SEC\\_POSITIONS\\_BY\\_INVESTMENTS](#)

## FV\_T\_FUND\_SEC\_POSITIONS\_BY\_INVESTMENTS

INVESTMENT\_ID;ID investment{F}ID Investissement

SHAREINDEX;  
FUND\_ID;ID fund{F}ID Fonds  
FUND\_NAME;Fund name{F}Nom du fonds  
FUND\_CURRENCY;Fund currency{F}Devise fonds  
FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
INVESTMENT\_NAME;Investment name{F}Nom investissement  
INVESTMENT\_TYPE\_CODE;Type  
INVESTMENT\_TYPE;Type  
INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement  
INVESTOR\_ID;ID investor{F}ID investisseur  
INVESTOR\_NAME;Investor name{F}Nom investisseur  
INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur  
SHARE\_ID;Share ID{F}ID Part  
SECURITY\_FUND\_ID;ID Fund share{F}ID Part  
SECURITY\_ID;Security ID{F}ID part  
SECURITY\_CURRENCY;Share currency{F}Devise part  
SECURITY\_PRICE1;Price{F}Nominal  
SECURITY\_ISCARRIED;Carried share{F}Part de carried  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
ID\_CURRTABLE\_FUND;  
NBSHARE\_ROUNDING;Number of shares rounding{F}Arrondi du nombre de parts  
NBSHARE\_ROUNDING\_TYPE;Number of share rounding mode  
SHAREPRICE\_ROUNDING;Share price rounding{F}Arrondi du nominal  
SHAREPRICE\_ROUNDING\_TYPE;Share price rounding mode{F}Type d'arrondi par part  
IS\_ROUNDINGSHARE\_CLASS;Is rounding share class{F}Classe réceptionnant les arrondis  
COMMITMENT\_IN\_AMOUNT;Commitment in amount  
RATE\_INV\_TO\_SEC;  
RATE\_SEC\_TO\_INV;  
RATE\_SEC\_TO\_FUND;  
SECURITY\_NB SHARES\_COMMITTED;Nb shares committed{F}Nb parts engagées  
SECURITY\_CALLED;Security called{F}Appelé part  
SECURITY\_CALLED1;Security called(1){F}Appelé part(1)

SECURITY\_CALLED2;Security called(2){F}Appelé part(2)  
SECURITY\_DISTRIBUTED;Security distributed{F}Distribué part  
SECURITY\_DISTRIBUTED1;Security distributed(1){F}Distribué part(1)  
SECURITY\_DISTRIBUTED2;Security distributed(2){F}Distribué part(2)  
NB SHARES;Nb shares{F}Nombre de parts  
NB SHARES\_CALLED;Nb shares called{F}Nombre de parts appelées  
NB SHARES\_BOUGHTBACK;Nb shares bought back{F}Nombre de parts rachetées  
REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel  
REMAINING\_COMMITMENT1;Remaining commitment(1){F}Engagement résiduel(1)  
REMAINING\_COMMITMENT2;Remaining commitment(2){F}Engagement résiduel(2)  
CAPITAL\_CALL;Capital call{F}Montant appelé  
CAPITAL\_CALL1;Capital call(1){F}Montant appelé(1)  
CAPITAL\_CALL2;Capital call(2){F}Montant appelé(2)  
CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement  
CAPITAL\_CALL\_INSIDE\_COMM1;Capital call inside commitment(1){F}Montant appelé dans l'engagement(1)  
CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)  
CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement  
CAPITAL\_CALL\_OUTSIDE\_COMM1;Capital call outside commitment(1){F}Montant appelé hors engagement(1)  
CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(2){F}Montant appelé hors engagement(2)  
MANAGEMENT\_FEES;Management fees{F}Frais de gestion  
MANAGEMENT\_FEES1;Management fees(1){F}Frais de gestion(1)  
MANAGEMENT\_FEES2;Management fees(2){F}Frais de gestion(2)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement  
MANAGEMENT\_FEES\_OUTSIDE\_COMM1;Management fees outside commitment(1){F}Frais de gestion hors engagement(1)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(2){F}Frais de gestion hors engagement(2)  
OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES1;Other fees(1){F}Autres frais(1)  
OTHER\_FEES2;Other fees(2){F}Autres frais(2)  
OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement  
OTHER\_FEES\_OUTSIDE\_COMM1;Other fees outside commitment(1){F}Autres frais hors engagement(1)  
OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(2){F}Autres frais hors engagement(2)  
DISTRIBUTION;Distributions{F}Distributions  
DISTRIBUTION1;Distributions(1){F}Distributions(1)  
DISTRIBUTION2;Distributions(2){F}Distributions(2)  
RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé  
RETURN\_OF\_CAPITAL1;Return of capital(1){F}Capital remboursé(1)  
RETURN\_OF\_CAPITAL2;Return of capital(2){F}Capital remboursé(2)  
REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire  
REDRAWABLE\_CAPITAL1;Redrawable call(1){F}Distribution temporaire(1)  
REDRAWABLE\_CAPITAL2;Redrawable call(2){F}Distribution temporaire(2)  
CARRY;Carry{F}Intérêt prioritaire  
CARRY1;Carry(1){F}Intérêt prioritaire(1)  
CARRY2;Carry(2){F}Intérêt prioritaire(2)  
GAIN\_LOSS;Gain/Loss{F}Plus/moins value  
GAIN\_LOSS1;Gain/Loss(1){F}Plus/moins value(1)  
GAIN\_LOSS2;Gain/Loss(2){F}Plus/moins value(2)  
INTERESTS;Interests{F}Intérêts  
INTERESTS1;Interests(1){F}Intérêts(1)  
INTERESTS2;Interests(2){F}Intérêts(2)  
DIVIDENDS;Dividends{F}Dividendes  
DIVIDENDS1;Dividends(1){F}Dividendes(1)  
DIVIDENDS2;Dividends(2){F}Dividendes(2)  
OTHER\_INCOME;Other income{F}Autres revenus  
OTHER\_INCOME1;Other income(1){F}Autres revenus(1)  
OTHER\_INCOME2;Other income(2){F}Autres revenus(2)  
TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions  
TRANSACTION\_COSTS1;Transaction costs(1){F}Coûts des transactions(1)  
TRANSACTION\_COSTS2;Transaction costs(2){F}Coûts des transactions(2)

COST;Cost{F}Prix de revient  
COST1;Cost(1){F}Prix de revient(1)  
COST2;Cost(2){F}Prix de revient(2)  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement  
GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif  
UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée  
TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
SECURITY\_NAME;Security name{F}Nom part  
SECURITY\_VALUE1;Security value(1){F}Valeur part(1)  
LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)  
LAST\_FUNDAMIN\_NOMINAL\_VALUATION1;Last valuation (fund admin)(1){F}Dernière valorisation (admin de fonds)(1)  
LAST\_FUNDAMIN\_NB SHARES;Nb shares for last valuation(fund admin){F}Nombre de parts à la dernière valo(admin de fonds)  
LAST\_FUNDAMIN\_ROUNDED\_VALUATION1;Last rounded total valuation (fund admin)(1){F}Dernière valorisation totale arrondie (admin de fonds)(1)  
LAST\_FUNDAMIN\_ROUNDED\_VALUATION;Last rounded total valuation (fund admin){F}Dernière valorisation totale arrondie (admin de fonds)  
LAST\_FUNDAMIN\_ROUNDED\_VALUATION2;Last rounded total valuation (fund admin)(2){F}Dernière valorisation totale arrondie (admin de fonds)(2)  
SECURITY\_NOMINAL\_CALLED;Nominal called{F}Nominal appelé  
SECURITY\_NOMINAL\_CALLED1;Nominal called(1){F}Nominal appelé(1)  
SECURITY\_NOMINAL\_BOUGHTBACK;Nominal bought{F}Nominal racheté  
SECURITY\_NOMINAL\_BOUGHTBACK1;Nominal bought(1){F}Nominal racheté(1)

LAST\_FUNDADMIN\_NOMINAL\_ROUNDED\_VALUATION1;Last rounded nominal valuation (fund admin)(1){F}Dernière valorisation nominale arrondie (admin de fonds)  
(1)

LAST\_FUNDADMIN\_NOMINAL\_ROUNDED\_VALUATION;Last rounded nominal valuation (fund admin){F}Dernière valorisation nominale arrondie (admin de fonds)

LAST\_FUNDADMIN\_NOMINAL\_ROUNDED\_VALUATION2;Last rounded nominal valuation (fund admin)(2){F}Dernière valorisation nominale arrondie (admin de fonds)  
(2)

COMMITMENT;Total commitment{F}Engagement total

COMMITMENT1;Total commitment(1){F}Engagement total(1)

COMMITMENT2;Total commitment(2){F}Engagement total(2)

SECURITY\_PRICE;Price{F}Nominal

SECURITY\_PRICE2;Price(2){F}Nominal(2)

SECURITY\_VALUE;Security value{F}Valeur part

SECURITY\_VALUE2;Security value(2){F}Valeur part(2)

SECURITY\_TOTAL\_COMMITMENT;Share total commitment{F}Engagement total part

SECURITY\_TOTAL\_COMMITMENT1;Share total commitment(1){F}Engagement total part(1)

SECURITY\_TOTAL\_COMMITMENT2;Share total commitment(2){F}Engagement total part(2)

SECURITY\_TOTAL\_VALUE;Share total value{F}Valeur totale part

SECURITY\_TOTAL\_VALUE1;Share total value(1){F}Valeur totale part(1)

SECURITY\_TOTAL\_VALUE2;Share total value(2){F}Valeur totale part(2)

SECURITY\_NOMINAL;Current nominal{F}Nominal courant

SECURITY\_NOMINAL1;Current nominal(1){F}Nominal courant(1)

SECURITY\_NOMINAL2;Current nominal(2){F}Nominal courant(2)

LAST\_FUNDADMIN\_NOMINAL\_VALUATION;Last valuation (fund admin){F}Dernière valorisation (admin de fonds)

LAST\_FUNDADMIN\_NOMINAL\_VALUATION2;Last valuation (fund admin)(2){F}Dernière valorisation (admin de fonds)(2)

SECURITY\_NB SHARES COMMITTED\_ROUNDED;Committed securities number{F}Nb parts engagées

NB\_SHARES\_ROUNDED;Nb shares{F}Nombre de parts

NB\_SHARES\_CALLED\_ROUNDED;Nb shares called{F}Nombre de parts appelées

- NB SHARES BOUGHTBACK\_ROUNDED;Nb shares bought back{F}Nombre de parts rachetées
- NETIRR\_BY\_SECURITY;Net IRR by Investor security{F}TRI NET par part investisseur
- LAST\_NAV\_DATE;Date{F}Date(first)
- ► [\*\*FV\\_T\\_FUND\\_SEC\\_POSITIONS\\_BY\\_FUNDS\*\*](#)

## **FV\_T\_FUND\_SEC\_POSITIONS\_BY\_FUNDS**

INVESTMENT\_ID;ID investment{F}ID Investissement  
SHAREINDEX;  
FUND\_ID;ID fund{F}ID Fonds  
FUND\_NAME;Fund name{F}Nom du fonds  
FUND\_CURRENCY;Fund currency{F}Devise fonds  
FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
INVESTMENT\_NAME;Investment name{F}Nom investissement  
INVESTMENT\_TYPE\_CODE;Type  
INVESTMENT\_TYPE;Type  
INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement  
INVESTOR\_ID;ID investor{F}ID investisseur  
INVESTOR\_NAME;Investor name{F}Nom investisseur  
INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur  
SHARE\_ID;Share ID{F}ID Part  
SECURITY\_FUND\_ID;ID Fund share{F}ID Part  
SECURITY\_ID;Security ID{F}ID part  
SECURITY\_CURRENCY;Share currency{F}Devise part  
SECURITY\_PRICE1;Price{F}Nominal  
SECURITY\_ISCARRIED;Carried share{F}Part de carried  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
ID\_CURRTABLE\_FUND;  
NBSHARE\_ROUNDING;Number of shares rounding{F}Arrondi du nombre de parts  
NBSHARE\_ROUNDING\_TYPE;Number of share rounding mode  
SHAREPRICE\_ROUNDING;Share price rounding{F}Arrondi du nominal

SHAREPRICE\_ROUNDING\_TYPE;Share price rounding mode{F}Type d'arrondi par part  
IS\_ROUNDINGSHARE\_CLASS;Is rounding share class{F}Classe réceptionnant les arrondis  
COMMITMENT\_IN\_AMOUNT;Commitment in amount  
RATE\_INV\_TO\_SEC;  
RATE\_SEC\_TO\_INV;  
RATE\_SEC\_TO\_FUND;  
SECURITY\_NB SHARES COMMITTED;Nb shares committed{F}Nb parts engagées  
SECURITY\_CALLED;Security called{F}Appelé part  
SECURITY\_CALLED1;Security called(1){F}Appelé part(1)  
SECURITY\_CALLED2;Security called(2){F}Appelé part(2)  
SECURITY\_DISTRIBUTED;Security distributed{F}Distribué part  
SECURITY\_DISTRIBUTED1;Security distributed(1){F}Distribué part(1)  
SECURITY\_DISTRIBUTED2;Security distributed(2){F}Distribué part(2)  
NB SHARES;Nb shares{F}Nombre de parts  
NB SHARES CALLED;Nb shares called{F}Nombre de parts appelées  
NB SHARES BOUGHTBACK;Nb shares bought back{F}Nombre de parts rachetées  
REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel  
REMAINING\_COMMITMENT1;Remaining commitment(1){F}Engagement résiduel(1)  
REMAINING\_COMMITMENT2;Remaining commitment(2){F}Engagement résiduel(2)  
CAPITAL\_CALL;Capital call{F}Montant appelé  
CAPITAL\_CALL1;Capital call(1){F}Montant appelé(1)  
CAPITAL\_CALL2;Capital call(2){F}Montant appelé(2)  
CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement  
CAPITAL\_CALL\_INSIDE\_COMM1;Capital call inside commitment(1){F}Montant appelé dans l'engagement(1)  
CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)  
CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement  
CAPITAL\_CALL\_OUTSIDE\_COMM1;Capital call outside commitment(1){F}Montant appelé hors engagement(1)

CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(2){F}Montant appelé hors engagement(2)  
MANAGEMENT\_FEES;Management fees{F}Frais de gestion  
MANAGEMENT\_FEES1;Management fees(1){F}Frais de gestion(1)  
MANAGEMENT\_FEES2;Management fees(2){F}Frais de gestion(2)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement  
MANAGEMENT\_FEES\_OUTSIDE\_COMM1;Management fees outside commitment(1){F}Frais de gestion hors engagement(1)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(2){F}Frais de gestion hors engagement(2)  
OTHER\_FEES;Other fees{F}Autres frais  
OTHER\_FEES1;Other fees(1){F}Autres frais(1)  
OTHER\_FEES2;Other fees(2){F}Autres frais(2)  
OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement  
OTHER\_FEES\_OUTSIDE\_COMM1;Other fees outside commitment(1){F}Autres frais hors engagement(1)  
OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(2){F}Autres frais hors engagement(2)  
DISTRIBUTION;Distributions{F}Distributions  
DISTRIBUTION1;Distributions(1){F}Distributions(1)  
DISTRIBUTION2;Distributions(2){F}Distributions(2)  
RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé  
RETURN\_OF\_CAPITAL1;Return of capital(1){F}Capital remboursé(1)  
RETURN\_OF\_CAPITAL2;Return of capital(2){F}Capital remboursé(2)  
REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire  
REDRAWABLE\_CAPITAL1;Redrawable call(1){F}Distribution temporaire(1)  
REDRAWABLE\_CAPITAL2;Redrawable call(2){F}Distribution temporaire(2)  
CARRY;Carry{F}Intérêt prioritaire  
CARRY1;Carry(1){F}Intérêt prioritaire(1)  
CARRY2;Carry(2){F}Intérêt prioritaire(2)  
GAIN\_LOSS;Gain/Loss{F}Plus/moins value  
GAIN\_LOSS1;Gain/Loss(1){F}Plus/moins value(1)  
GAIN\_LOSS2;Gain/Loss(2){F}Plus/moins value(2)

INTERESTS;Interests{F}Intérêts  
INTERESTS1;Interests(1){F}Intérêts(1)  
INTERESTS2;Interests(2){F}Intérêts(2)  
DIVIDENDS;Dividends{F}Dividendes  
DIVIDENDS1;Dividends(1){F}Dividendes(1)  
DIVIDENDS2;Dividends(2){F}Dividendes(2)  
OTHER\_INCOME;Other income{F}Autres revenus  
OTHER\_INCOME1;Other income(1){F}Autres revenus(1)  
OTHER\_INCOME2;Other income(2){F}Autres revenus(2)  
TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions  
TRANSACTION\_COSTS1;Transaction costs(1){F}Coûts des transactions(1)  
TRANSACTION\_COSTS2;Transaction costs(2){F}Coûts des transactions(2)  
COST;Cost{F}Prix de revient  
COST1;Cost(1){F}Prix de revient(1)  
COST2;Cost(2){F}Prix de revient(2)  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement  
GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intérêts hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif  
UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée  
TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
SECURITY\_NAME;Security name{F}Nom part  
SECURITY\_VALUE1;Security value(1){F}Valeur part(1)  
LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)  
LAST\_FUNDAMIN\_NOMINAL\_VALUATION1;Last valuation (fund admin)(1){F}Dernière valorisation (admin de fonds)(1)

LAST\_FUNDAMIN\_NB SHARES;Nb shares for last valuation(fund admin){F}Nombre de parts à la dernière valo(admin de fonds)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION1;Last rounded total valuation (fund admin)(1){F}Dernière valorisation totale arrondie (admin de fonds)(1)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION;Last rounded total valuation (fund admin){F}Dernière valorisation totale arrondie (admin de fonds)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION2;Last rounded total valuation (fund admin)(2){F}Dernière valorisation totale arrondie (admin de fonds)(2)

SECURITY\_NOMINAL\_CALLED;Nominal called{F}Nominal appelé

SECURITY\_NOMINAL\_CALLED1;Nominal called(1){F}Nominal appelé(1)

SECURITY\_NOMINAL\_BOUGHTBACK;Nominal bought{F}Nominal racheté

SECURITY\_NOMINAL\_BOUGHTBACK1;Nominal bought(1){F}Nominal racheté(1)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION1;Last rounded nominal valuation (fund admin)(1){F}Dernière valorisation nominale arrondie (admin de fonds)(1)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION;Last rounded nominal valuation (fund admin){F}Dernière valorisation nominale arrondie (admin de fonds)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION2;Last rounded nominal valuation (fund admin)(2){F}Dernière valorisation nominale arrondie (admin de fonds)(2)

COMMITMENT;Total commitment{F}Engagement total

COMMITMENT1;Total commitment(1){F}Engagement total(1)

COMMITMENT2;Total commitment(2){F}Engagement total(2)

SECURITY\_PRICE;Price{F}Nominal

SECURITY\_PRICE2;Price(2){F}Nominal(2)

SECURITY\_VALUE;Security value{F}Valeur part

SECURITY\_VALUE2;Security value(2){F}Valeur part(2)

SECURITY\_TOTAL\_COMMITMENT;Share total commitment{F}Engagement total part

SECURITY\_TOTAL\_COMMITMENT1;Share total commitment(1){F}Engagement total part(1)

SECURITY\_TOTAL\_COMMITMENT2;Share total commitment(2){F}Engagement total part(2)

SECURITY\_TOTAL\_VALUE;Share total value{F}Valeur totale part

SECURITY\_TOTAL\_VALUE1;Share total value(1){F}Valeur totale part(1)

SECURITY\_TOTAL\_VALUE2;Share total value(2){F}Valeur totale part(2)

- SECURITY\_NOMINAL;Current nominal{F}Nominal courant  
SECURITY\_NOMINAL1;Current nominal(1){F}Nominal courant(1)  
SECURITY\_NOMINAL2;Current nominal(2){F}Nominal courant(2)  
LAST\_FUNDAMIN\_NOMINAL\_VALUATION;Last valuation (fund admin){F}Dernière valorisation (admin de fonds)  
LAST\_FUNDAMIN\_NOMINAL\_VALUATION2;Last valuation (fund admin)(2){F}Dernière valorisation (admin de fonds)(2)  
SECURITY\_NB SHARES\_COMMITED\_ROUNDED;Committed securities number{F}Nb parts engagées  
NB\_SHARES\_ROUNDED;Nb shares{F}Nombre de parts  
NB\_SHARES\_CALLED\_ROUNDED;Nb shares called{F}Nombre de parts appelées  
NB\_SHARES\_BOUGHTBACK\_ROUNDED;Nb shares bought back{F}Nombre de parts rachetées  
NET\_IRR\_BY\_SECURITY;Net IRR by Investor security{F}TRI NET par part investisseur  
LAST\_NAV\_DATE;Date{F}Date(first)
- ▶ [FV\\_T\\_FUND\\_POSITIONS\\_BY\\_FUNDS](#)

## FV\_T\_FUND\_POSITIONS\_BY\_FUNDS

FUND\_ID;ID fund{F}ID Fonds  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel  
CASH\_IN;Cash in{F}Cash in  
CASH\_OUT;Cash out{F}Cash out  
CAPITAL\_CALL;Capital call{F}Montant appelé  
CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement  
CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement  
MANAGEMENT\_FEES;Management fees{F}Frais de gestion  
MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement  
OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement  
DISTRIBUTION;Distributions{F}Distributions  
RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé  
REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire  
CARRY;Carry{F}Intérêt prioritaire  
GAIN\_LOSS;Gain/Loss{F}Plus/moins value  
INTERESTS;Interests{F}Intérêts  
DIVIDENDS;Dividends{F}Dividendes  
OTHER\_INCOME;Other income{F}Autres revenus  
TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions  
COST;Cost{F}Prix de revient  
ADJUSTED\_NAV;Adjusted NAV{F}NAV ajustée  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenues hors engagement  
GAIN LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif  
UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée  
TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
LAST\_NAV;Last NAV{F}Dernière NAV  
LAST\_NAV\_DATE;Last NAV date{F}Dernière date NAV  
COMMITMENT;Total commitment{F}Engagement total  
CASH\_ON\_CASH;Cash on cash{F}Cash on cash  
CASH\_BALANCE;Cash balance{F}Balance CASH  
NAV\_ADJUSTMENT;NAV adjustment{F}Ajustement NAV  
PERCENT\_CALLED;% called{F}% appelé  
TVPI;TVPI{F}TVPI

DPI;DPI{F}DPI  
RVPI;RVPI{F}RVPI  
NET\_IRR;Net IRR{F}TRI NET  
FUND\_NAME;Fund{F}Fonds  
FUND\_ABBREVIATION;Short name{F}Abréviation  
FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
FUND\_STATUS;Status{F}Statut  
FUND\_STATUS\_FORMATTED;Status  
FUND\_TYPE;Legal form  
FUND\_REGION;Geography  
MANAGEMENT\_COMPANY\_ID;IQId  
MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion  
VINTAGE\_YEAR;Vintage year{F}Millésime  
FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing  
FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing  
CURRENCY;Currency{F}Devise  
DOMICILE;Domicile{F}Domiciliation  
LEGAL\_FORM;Legal form{F}Forme juridique  
FUND\_SIZE;Size{F}Taille  
TERM\_YEARS;Term (Years){F}Durée (années)  
INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)  
TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable  
END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement  
INVESTMENT\_PERIOD\_ADD;Extension  
EVCA\_STRUCTURE;Structure  
EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage  
EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography  
FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV  
LAST\_FUNDAMIN\_VALUATION;Last total valuation (fund admin){F}Dernière valorisation totale (admin de fonds)  
LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)  
NET\_IRR\_WO\_CARRIED;Net IRR wo Carried{F}TRI NET Hors Carried

- ▶ [FV\\_T\\_FUND\\_POSITIONS\\_BY\\_INVESTMENTS](#)

## FV\_T\_FUND\_POSITIONS\_BY\_FUNDS

FUND\_CURRENCY;Fund currency{F}Devise fonds  
INIT\_COMMITMENT;Initial commitment{F}Engagement initial  
SHAREINIT1;Share init 1{F}Titre init 1  
SHAREINIT2;Share init 2{F}Titre init 2  
SHAREINIT3;Share init 3{F}Titre init 3  
SHAREINIT4;Share init 4{F}Titre init 4  
SHAREINIT5;Share init 5{F}Titre init 5  
SHAREINIT6;Share init 6{F}Titre init 6  
SHAREINIT7;Share init 7{F}Titre init 7  
SHAREINIT8;Share init 8{F}Titre init 8  
SHAREINIT9;Share init 9{F}Titre init 9  
SHAREINIT10;Share init 10{F}Titre init 10  
SHAREINIT11;Share init 11{F}Titre init 11  
SHAREINIT12;Share init 12{F}Titre init 12  
SHAREINIT13;Share init 13{F}Titre init 13  
SHAREINIT14;Share init 14{F}Titre init 14  
SHAREINIT15;Share init 15{F}Titre init 15  
SHAREINIT16;Share init 16{F}Titre init 16  
SHAREINIT17;Share init 17{F}Titre init 17  
SHAREINIT18;Share init 18{F}Titre init 18  
SHAREINIT19;Share init 19{F}Titre init 19  
SHAREINIT20;Share init 20{F}Titre init 20  
INVESTMENT\_ID;ID investment{F}ID investissement  
INVESTMENT\_NAME;Investment name{F}Nom investissement  
INVESTMENT\_TYPE\_CODE;Type  
INVESTMENT\_TYPE;Type  
INVESTMENT\_CLOSING\_DATE;Investment closing date{F}Date de premier closing  
INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement  
INVESTMENT\_EXPL\_STAKE;Investment forced stake{F}% détention forcé  
INVESTMENT\_EXPL\_STAKE\_NC;Investment forced stake NC{F}% détention forcé  
NC

INVESTOR\_FUND\_ID;ID fund investor{F}ID fonds Investisseur  
INVESTOR\_FUND\_NAME;Name fund investor{F}Nom fonds Investisseur  
INVESTOR\_FUND\_CURRENCY;Currency fund investor{F}Devise fonds Investisseur  
INVESTOR\_FUND\_STATUS;Status fund investor{F}Statut fonds Investisseur  
INVESTOR\_COMPANY\_ID;ID company investor{F}ID société Investisseur  
INVESTOR\_COMPANY\_NAME;Name company investor{F}Nom société Investisseur  
INVESTOR\_COMPANY\_CURRENCY;Currency company investor{F}Devise société  
Investisseur  
INVESTOR\_COMPANY\_STATUS;Status company investor{F}Statut société  
Investisseur  
INVESTOR\_CONTACT\_ID;ID contact investor{F}ID contact Investisseur  
INVESTOR\_CONTACT\_LASTNAME;Lastname contact investor{F}Nom contact  
Investisseur  
INVESTOR\_CONTACT\_FIRSTNAME;Firstname contact investor{F}Prénom contact  
Investisseur  
INVESTOR\_CONTACT\_CURRENCY;Currency contact investor{F}Devise contact  
Investisseur  
INVESTOR\_CONTACT\_STATUS;Status contact investor{F}Statut contact Investisseur  
INVESTMENT\_CURRENCY;Currency(2)  
FUND\_ID;ID fund{F}ID Fonds  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé  
dans l'engagement  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of  
commitment{F}Autres revenus hors engagement  
GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of  
commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets  
hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions  
outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of  
commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée  
TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
COMMITMENT;Total commitment{F}Engagement total  
FUND\_NAME;Fund{F}Fonds  
FUND\_ABBREVIATION;Short name{F}Abréviation  
FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
FUND\_STATUS;Status{F}Statut  
FUND\_STATUS\_FORMATTED;Status  
FUND\_TYPE;Legal form  
FUND\_REGION;Geography  
MANAGEMENT\_COMPANY\_ID;IQId  
MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion  
VINTAGE\_YEAR;Vintage year{F}Millésime  
FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing  
FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing  
CURRENCY;Currency{F}Devise  
DOMICILE;Domicile{F}Domiciliation  
LEGAL\_FORM;Legal form{F}Forme juridique  
TOTAL\_FUND\_COMMITMENT;Size{F}Taille  
TERM\_YEARS;Term (Years){F}Durée (années)  
INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement  
(années)  
TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable  
END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement  
INVESTMENT\_PERIOD\_ADD;Extension  
EVCA\_STRUCTURE;Structure  
EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage  
EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography  
FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV  
LAST\_FUNDADMIN\_VALUATION;Last total valuation (fund admin){F}Dernière  
valorisation totale (admin de fonds)  
LAST\_FUNDADMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de  
dernière valorisation (admin de fonds)  
INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur  
INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur  
REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel  
REMAINING\_COMMITMENT2;Remaining commitment(Inv curr){F}Engagement résiduel(devise inv)  
CASH\_IN;Cash in{F}Cash in  
CASH\_IN2;Cash in(Inv curr){F}Cash in(devise inv)  
CASH\_OUT;Cash out{F}Cash out  
CASH\_OUT2;Cash out(Inv curr){F}Cash out(devise inv)  
CAPITAL\_CALL;Capital call{F}Montant appelé  
CAPITAL\_CALL2;Capital call(Inv curr){F}Montant appelé(devise inv)  
CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)  
CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement  
CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(Inv curr){F}Montant appelé hors engagement(devise inv)  
MANAGEMENT\_FEES;Management fees{F}Frais de gestion  
MANAGEMENT\_FEES2;Management fees(Inv curr){F}Frais de gestion(devise inv)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement  
MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(Inv curr){F}Frais de gestion hors engagement(devise inv)  
OTHER\_FEES;Other fees{F}Autres frais  
OTHER\_FEES2;Other fees(Inv curr){F}Autres frais(devise inv)  
OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement  
OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(Inv curr){F}Autres frais hors engagement(devise inv)  
DISTRIBUTION;Distributions{F}Distributions  
DISTRIBUTION2;Distributions(Inv curr){F}Distributions(devise inv)  
RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé  
RETURN\_OF\_CAPITAL2;Return of capital(Inv curr){F}Capital remboursé(devise inv)  
REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

REDRAWABLE\_CAPITAL2;Redrawable call(Inv curr){F}Distribution temporaire(devise inv)  
CARRY;Carry{F}Intérêt prioritaire  
CARRY2;Carry(Inv curr){F}Intérêt prioritaire(devise inv)  
GAIN\_LOSS;Gain/Loss{F}Plus/moins value  
GAIN\_LOSS2;Gain/Loss(Inv curr){F}Plus/moins value(devise inv)  
INTERESTS;Interests{F}Intérêts  
INTERESTS2;Interests(Inv curr){F}Intérêts(devise inv)  
DIVIDENDS;Dividends{F}Dividendes  
DIVIDENDS2;Dividends(Inv curr){F}Dividendes(devise inv)  
OTHER\_INCOME;Other income{F}Autres revenus  
OTHER\_INCOME2;Other income(Inv curr){F}Autres revenus(devise inv)  
TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions  
TRANSACTION\_COSTS2;Transaction costs(Inv curr){F}Coûts des transactions(devise inv)  
COST;Cost{F}Prix de revient  
COST2;Cost(2){F}Prix de revient(2)  
ADJUSTED\_NAV;Adjusted NAV{F}NAV ajustée  
ADJUSTED\_NAV2;Adjusted NAV(Inv curr){F}NAV ajustée(devise inv)  
EXPL\_STAKE;Explicit stake{F}% détention explicite  
EXPL\_STAKE\_NC;Explicit stake NC{F}% détention explicite NC  
LAST\_NAV;Last NAV{F}Dernière NAV  
LAST\_NAV\_DATE;Last NAV date{F}Dernière date NAV  
LAST\_FUNDAMIN\_VALUATION2;Last total valuation (fund admin)(2){F}Dernière valorisation totale (admin de fonds)(2)  
COMMITMENT2;Total commitment(Inv curr){F}Engagement total(devise inv)  
NAV\_ADJUSTMENT;NAV adjustment{F}Ajustement NAV  
CASH\_BALANCE;Cash balance{F}Balance CASH  
CASH\_BALANCE2;Cash balance(Inv curr){F}Balance CASH(devise inv)  
PERCENT\_CALLED;% called{F}% appelé  
PERCENT\_CALLED2;% called(Inv curr){F}% appelé(devise inv)  
TVPI;TVPI{F}TVPI  
DPI;DPI{F}DPI  
RVPI;RVPI{F}RVPI  
TVPI2;TVPI(Inv curr){F}TVPI(devise inv)

DPI2;DPI(Inv curr){F}DPI(devise inv)  
RVPI2;RVPI(Inv curr){F}RVPI(devise inv)  
NET\_IRR;Net IRR{F}TRI NET  
NET\_IRR2;Net IRR(Inv curr){F}TRI NET(devise inv)  
CASH\_ON\_CASH;Cash on cash{F}Cash on cash  
CASH\_ON\_CASH2;Cash on cash(Inv curr){F}Cash on cash(devise inv)  
STAKE;% Stake{F}% détention  
STAKE\_NC;%Stake{F}% détention  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement  
GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif  
UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée  
TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
LAST\_NAV;Last NAV{F}Dernière NAV  
LAST\_NAV\_DATE;Last NAV date{F}Dernière date NAV  
COMMITMENT;Total commitment{F}Engagement total  
CASH\_ON\_CASH;Cash on cash{F}Cash on cash  
CASH\_BALANCE;Cash balance{F}Balance CASH  
NAV\_ADJUSTMENT;NAV adjustment{F}Ajustement NAV  
PERCENT\_CALLED;% called{F}% appelé  
TVPI;TVPI{F}TVPI  
DPI;DPI{F}DPI  
RVPI;RVPI{F}RVPI  
NET\_IRR;Net IRR{F}TRI NET  
FUND\_NAME;Fund{F}Fonds  
FUND\_ABBREVIATION;Short name{F}Abréviation

FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
FUND\_STATUS;Status{F}Statut  
FUND\_STATUS\_FORMATTED;Status  
FUND\_TYPE;Legal form  
FUND\_REGION;Geography  
MANAGEMENT\_COMPANY\_ID;IQId  
MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion  
VINTAGE\_YEAR;Vintage year{F}Millésime  
FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing  
FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing  
CURRENCY;Currency{F}Devise  
DOMICILE;Domicile{F}Domiciliation  
LEGAL\_FORM;Legal form{F}Forme juridique  
FUND\_SIZE;Size{F}Taille  
TERM\_YEARS;Term (Years){F}Durée (années)  
INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)  
TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable  
END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement  
INVESTMENT\_PERIOD\_ADD;Extension  
EVCA\_STRUCTURE;Structure  
EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage  
EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography  
FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV  
LAST\_FUNDAMIN\_VALUATION;Last total valuation (fund admin){F}Dernière valorisation totale (admin de fonds)  
LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)  
NET\_IRR\_WO\_CARRIED;Net IRR wo Carried{F}TRI NET Hors Carried

- ► [FV\\_T\\_FUND\\_POSITIONS\\_BY\\_INVESTORS](#)

## FV\_T\_FUND\_POSITIONS\_BY\_INVESTORS

INVESTOR\_ID;ID investor{F}ID investisseur  
INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur  
INVESTOR\_FUND\_ID;ID fund investor{F}ID fonds Investisseur  
INVESTOR\_FUND\_NAME;Name fund investor{F}Nom fonds Investisseur  
INVESTOR\_FUND\_CURRENCY;Currency fund investor{F}Devise fonds Investisseur  
INVESTOR\_FUND\_STATUS;Status fund investor{F}Statut fonds Investisseur  
INVESTOR\_COMPANY\_ID;ID company investor{F}ID société Investisseur  
INVESTOR\_COMPANY\_NAME;Name company investor{F}Nom société Investisseur  
INVESTOR\_COMPANY\_CURRENCY;Currency company investor{F}Devise société Investisseur  
INVESTOR\_COMPANY\_STATUS;Status company investor{F}Statut société Investisseur  
INVESTOR\_CONTACT\_ID;ID contact investor{F}ID contact Investisseur  
INVESTOR\_CONTACT\_LASTNAME;Lastname contact investor{F}Nom contact Investisseur  
INVESTOR\_CONTACT\_FIRSTNAME;Firstname contact investor{F}Prénom contact Investisseur  
INVESTOR\_CONTACT\_CURRENCY;Currency contact investor{F}Devise contact Investisseur  
INVESTOR\_CONTACT\_STATUS;Status contact investor{F}Statut contact Investisseur  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
COMMITMENT2;Total commitment(Inv curr){F}Engagement total(devise inv)  
REMAINING\_COMMITMENT2;Remaining commitment(Inv curr){F}Engagement résiduel(devise inv)  
CASH\_IN2;Cash in(Inv curr){F}Cash in(devise inv)  
CASH\_OUT2;Cash out(Inv curr){F}Cash out(devise inv)  
CAPITAL\_CALL2;Capital call(Inv curr){F}Montant appelé(devise inv)  
CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)  
CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(Inv curr){F}Montant appelé hors engagement(devise inv)  
MANAGEMENT\_FEES2;Management fees(Inv curr){F}Frais de gestion(devise inv)  
MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(Inv curr){F}Frais de gestion hors engagement(devise inv)  
OTHER\_FEES2;Other fees(Inv curr){F}Autres frais(devise inv)

OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(Inv curr)  
{F}Autres frais hors engagement(devise inv)

DISTRIBUTION2;Distributions(Inv curr){F}Distributions(devise inv)

RETURN\_OF\_CAPITAL2;Return of capital(Inv curr){F}Capital remboursé(devise inv)

REDRAWABLE\_CAPITAL2;Redrawable call(Inv curr){F}Distribution temporaire(devise inv)

CARRY2;Carry(Inv curr){F}Intérêt prioritaire(devise inv)

GAIN\_LOSS2;Gain/Loss(Inv curr){F}Plus/moins value(devise inv)

INTERESTS2;Interests(Inv curr){F}Intérêts(devise inv)

DIVIDENDS2;Dividends(Inv curr){F}Dividendes(devise inv)

OTHER\_INCOME2;Other income(Inv curr){F}Autres revenus(devise inv)

TRANSACTION\_COSTS2;Transaction costs(Inv curr){F}Coûts des transactions(devise inv)

COST2;Cost(2){F}Prix de revient(2)

ADJUSTED\_NAV2;NAV(Inv curr){F}NAV(devise inv)

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

CASH\_BALANCE2;Cash balance(Inv curr){F}Balance CASH(devise inv)

CASH\_ON\_CASH2;Cash on cash{F}Cash on cash

PERCENT\_CALLED2;% called(Inv curr){F}% appelé(devise inv)

TVPI2;TVPI(Inv curr){F}TVPI(devise inv)

DPI2;DPI(Inv curr){F}DPI(devise inv)

RVPI2;RVPI(Inv curr){F}RVPI(devise inv)

- ► [FV\\_T\\_FUND\\_OPERATIONS\\_BY\\_INVESTORS](#)

## FV\_T\_FUND\_OPERATIONS\_BY\_INVESTORS

OPERATION\_ID;ID fund operation{F}ID opération fonds  
INVESTMENT\_ID;ID investment{F}ID Investissement  
DRAFT;Draft{F}Brouillon  
TRANSACTION\_CANCELLED;Cancelled  
FUND\_ID;ID fund{F}ID Fonds  
TYPE;Type{F}Type  
OPERATION\_TYPE;Formatted type{F}Type formatté  
OPERATION\_NAME;Name fund operation{F}Libellé opération fonds  
YEAR;Year{F}Année  
REFERENCE\_DATE;Date{F}Date  
INDEX;Index{F}Index  
INVESTOR\_ID;ID investor{F}ID investisseur  
INVESTOR\_NAME;Investor name{F}Nom investisseur  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
COMMITMENT;Commitment{F}Engagement  
CASH\_IN;Cash IN{F}Cash IN  
CASH\_OUT;Cash OUT{F}Cash OUT  
CAPITAL\_CALL;Capital call{F}Montant appelé  
CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement  
CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement  
MANAGEMENT\_FEES;Management fees{F}Frais de gestion  
MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement  
OTHER\_FEES;Other fees{F}Autres frais  
OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement  
DISTRIBUTION;Distributions{F}Distributions

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé  
REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire  
CARRY;Carry{F}Intérêt prioritaire  
GAIN\_LOSS;Gain/Loss{F}Plus/moins value  
INTERESTS;Interests{F}Intérêts  
DIVIDENDS;Dividends{F}Dividendes  
OTHER\_INCOME;Other income{F}Autres revenus  
TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions  
COST;Cost{F}Prix de revient  
NAV;NAV{F}NAV  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement  
GAIN LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intérêts hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif  
UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée  
TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
CASH\_BALANCE;Cash balance{F}Balance CASH  
CASH\_ON\_CASH;Cash on cash{F}Cash on cash  
FUND\_NAME;Fund{F}Fonds  
FUND\_ABBREVIATION;Short name{F}Abréviation  
FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
FUND\_STATUS;Status{F}Statut  
FUND\_STATUS\_FORMATTED;Status  
FUND\_TYPE;Legal form  
FUND\_REGION;Geography  
MANAGEMENT\_COMPANY\_ID;IQId  
MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion

VINTAGE\_YEAR;Vintage year{F}Millésime  
FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing  
FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing  
CURRENCY;Currency{F}Devise  
DOMICILE;Domicile{F}Domiciliation  
LEGAL\_FORM;Legal form{F}Forme juridique  
FUND\_SIZE;Size{F}Taille  
TERM\_YEARS;Term (Years){F}Durée (années)  
INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)  
TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable  
END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement  
INVESTMENT\_PERIOD\_ADD;Extension  
EVCA\_STRUCTURE;Structure  
EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage  
EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography  
FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV

- ► [FV\\_T\\_FUND\\_OPERATIONS\\_BY\\_FUNDS](#)

## FV\_T\_FUND\_OPERATIONS\_BY\_FUNDS

OPERATION\_ID;ID fund operation{F}ID opération fonds  
DRAFT;Draft{F}Brouillon  
TRANSACTION\_CANCELLED;Cancelled  
FUND\_ID;ID fund{F}ID Fonds  
TYPE;Type{F}Type  
OPERATION\_TYPE;Formatted type{F}Type formatté  
OPERATION\_NAME;Name fund operation{F}Libellé opération fonds  
YEAR;Year{F}Année  
REFERENCE\_DATE;Date{F}Date  
INDEX;Index{F}Index  
REPORT\_DATE;As of Date{F}Date d'arrêté  
DATAMART\_DATE;Datamart date{F}Date du datamart  
COMMITMENT;Commitment{F}Engagement  
CASH\_IN;Cash IN{F}Cash IN

CASH\_OUT;Cash OUT{F}Cash OUT  
CAPITAL\_CALL;Capital call{F}Montant appelé  
CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement  
CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement  
MANAGEMENT\_FEES;Management fees{F}Frais de gestion  
MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement  
OTHER\_FEES;Other fees{F}Autres frais  
OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement  
DISTRIBUTION;Distributions{F}Distributions  
RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé  
REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire  
CARRY;Carry{F}Intérêt prioritaire  
GAIN\_LOSS;Gain/Loss{F}Plus/moins value  
INTERESTS;Interests{F}Intérêts  
DIVIDENDS;Dividends{F}Dividendes  
OTHER\_INCOME;Other income{F}Autres revenus  
TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions  
COST;Cost{F}Prix de revient  
NAV;NAV{F}NAV  
OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement  
GAIN LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement  
INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement  
UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement  
LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement  
LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif  
UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe  
TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé  
CASH\_BALANCE;Cash balance{F}Balance CASH  
FUND\_NAME;Fund{F}Fonds  
FUND\_ABBREVIATION;Short name{F}Abréviation  
FUND\_GROUPBY\_ID;Master fund{F}Fonds master  
FUND\_STATUS;Status{F}Statut  
FUND\_STATUS\_FORMATTED;Status  
FUND\_TYPE;Legal form  
FUND\_REGION;Geography  
MANAGEMENT\_COMPANY\_ID;IQId  
MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion  
VINTAGE\_YEAR;Vintage year{F}Millésime  
FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing  
FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing  
CURRENCY;Currency{F}Devise  
DOMICILE;Domicile{F}Domiciliation  
LEGAL\_FORM;Legal form{F}Forme juridique  
FUND\_SIZE;Size{F}Taille  
TERM\_YEARS;Term (Years){F}Durée (années)  
INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)  
TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable  
END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement  
INVESTMENT\_PERIOD\_ADD;Extension  
EVCA\_STRUCTURE;Structure  
EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage  
EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography  
FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV

## 6.9.1. FundOperations

### 6.9.1.1. FUNDOPERATIONS

New CECubeColumn() { \_

New CECubeColumn("FUND\_ID", "Fund ID{F}ID du Fonds", GetType(String)), \_  
New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)), \_  
—  
New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds", GetType(String)), \_  
New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des montants", GetType(String)), \_  
New CECubeColumn("OPERATION\_ID", "Operation ID{F}ID opération", GetType(String)), \_  
New CECubeColumn("OPERATIONTYPE", "Operation type{F}Type opération", GetType(Int32)), \_  
New CECubeColumn("OPERATIONSUBTYPE", "Operation subtype{F}Sous-type opération", GetType(Int32)), \_  
New CECubeColumn("OPERATIONDATE", "Operation date{F}Date opération", GetType(DateTime)), \_  
New CECubeColumn("OPERATIONNAME", "Operation name{F}Nom opération", GetType(String)), \_  
New CECubeColumn("COMMITMENT", "Commitment{F}Engagement", GetType(Decimal)), \_  
New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_  
New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_  
New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_  
New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_  
New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

```
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _  
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",  
GetType(Decimal)), _  
New CECubeColumn("VALUATION", "NAV{F}NAV", GetType(Decimal)), _  
New CECubeColumn("VALUATIONDELTA", "Valuation delta{F}Delta valorisation",  
GetType(Decimal)), _  
New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",  
GetType(Decimal)), _  
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",  
GetType(DateTime)) _  
}
```

## 6.9.2. FundShareOperations

### 6.9.2.1. FUNDSHAREOPERATIONS

```
New CECubeColumn() { _  
New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _  
New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)), _  
—  
New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",  
GetType(String)), _  
New CECubeColumn("SHARE_ID", "Share ID{F}ID part", GetType(String)), _  
New CECubeColumn("SHAREINDEX", "Share index{F}Index part", GetType(String)), _  
New CECubeColumn("SHARECURRENCY", "Share currency{F}Devise de la part",  
GetType(String)), _  
New CECubeColumn("SHARENAME", "Share name{F}Nom de la part",  
GetType(String)), _
```

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des montants", GetType(String)), \_

New CECubeColumn("OPERATION\_ID", "Operation ID{F}ID opération", GetType(String)), \_

New CECubeColumn("OPERATIONTYPE", "Operation type{F}Type opération", GetType(Int32)), \_

New CECubeColumn("OPERATIONSUBTYPE", "Operation subtype{F}Sous-type opération", GetType(Int32)), \_

New CECubeColumn("OPERATIONDATE", "Operation date{F}Date opération", GetType(DateTime)), \_

New CECubeColumn("OPERATIONNAME", "Operation name{F}Nom opération", GetType(String)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement", GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus", GetType(Decimal)), \_

New CECubeColumn("VALUATION", "NAV{F}NAV", GetType(Decimal)), \_

```
New CECubeColumn("VALUATIONDELTA", "Valuation delta{F}Delta valorisation",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",
GetType(DateTime)) _

}, _
```

### 6.9.3. FundInvestorOperations

#### 6.9.3.1. FUNDINVESTOROPERATIONS

```
New CECubeColumn() { _

New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _

New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)),
_

New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",
GetType(String)), _

New CECubeColumn("INVESTMENT_ID", "Subscriber ID{F}ID du souscripteur",
GetType(String)), _

New CECubeColumn("INVESTMENTNAME", "Subscriber name{F}Nom du
souscripteur", GetType(String)), _

New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",
GetType(String)), _
```

New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de l'investisseur", GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des montants", GetType(String)), \_

New CECubeColumn("OPERATION\_ID", "Operation ID{F}ID opération", GetType(String)), \_

New CECubeColumn("OPERATIONTYPE", "Operation type{F}Type opération", GetType(Int32)), \_

New CECubeColumn("OPERATIONSUBTYPE", "Operation subtype{F}Sous-type opération", GetType(Int32)), \_

New CECubeColumn("OPERATIONDATE", "Operation date{F}Date opération", GetType(DateTime)), \_

New CECubeColumn("OPERATIONNAME", "Operation name{F}Nom opération", GetType(String)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement", GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

```
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
GetType(Decimal)), _

New CECubeColumn("VALUATION", "NAV{F}NAV", GetType(Decimal)), _

New CECubeColumn("VALUATIONDELTA", "Valuation delta{F}Delta valorisation",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",
GetType(DateTime)) _

}
```

## 6.9.4. FundInvestorShareOperations

### 6.9.4.1. FUNDINVESTORSHAREOPERATIONS

```
New CECubeColumn() {_

New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _

New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)),
_

New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",
GetType(String)), _

New CECubeColumn("SHARE_ID", "Share ID{F}ID part", GetType(String)), _

New CECubeColumn("SHAREINDEX", "Share index{F}Index part", GetType(String)), _

New CECubeColumn("SHARECURRENCY", "Share currency{F}Devise de la part",
GetType(String)), _

New CECubeColumn("SHARENAME", "Share name{F}Nom de la part",
GetType(String)), _
```

New CECubeColumn("INVESTMENT\_ID", "Subscriber ID{F}ID du souscripteur",  
GetType(String)), \_

New CECubeColumn("INVESTMENTNAME", "Subscriber name{F}Nom du  
souscripteur", GetType(String)), \_

New CECubeColumn("INVESTOR\_ID", "Investor ID{F}ID de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de  
l'investisseur", GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des  
montants", GetType(String)), \_

New CECubeColumn("OPERATION\_ID", "Operation ID{F}ID opération",  
GetType(String)), \_

New CECubeColumn("OPERATIONTYPE", "Operation type{F}Type opération",  
GetType(Int32)), \_

New CECubeColumn("OPERATIONSUBTYPE", "Operation subtype{F}Sous-type  
opération", GetType(Int32)), \_

New CECubeColumn("OPERATIONDATE", "Operation date{F}Date opération",  
GetType(DateTime)), \_

New CECubeColumn("OPERATIONNAME", "Operation name{F}Nom opération",  
GetType(String)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",  
GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

```
New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital",
GetType(Decimal)), _

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital
rappelable", GetType(Decimal)), _

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",
GetType(Decimal)), _

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _

New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
GetType(Decimal)), _

New CECubeColumn("VALUATION", "NAV{F}NAV", GetType(Decimal)), _

New CECubeColumn("VALUATIONDELTA", "Valuation delta{F}Delta valorisation",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",
GetType(DateTime)) _

}
```

## 6.9.5. FundPositions and FundPositionsWithoutCarried

### 6.9.5.1. FUNDPOSITIONS / FUNDPOSITIONSWITHOUTCARRIED

```
New CECubeColumn() { _
```

```
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",
GetType(DateTime)), _
```

```
New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _
```

New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)), \_

New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds", GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des montants", GetType(String)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement", GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("PERCENTCALLED", "% called{F}% appelé", GetType(Double)), \_

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus", GetType(Decimal)), \_

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV", GetType(Decimal)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV", GetType(DateTime)), \_

```
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",
GetType(Decimal)), _

New CECubeColumn("IRRNET", "Net IRR{F}TRI net", GetType(Double)), _

New CECubeColumn("IRRNETREALIZED", "Realized net IRR{F}TRI net cédé",
GetType(Double)), _

New CECubeColumn("IRRNETUNREALIZED", " Unrealized net IRR{F}TRI net non
cédé", GetType(Double)), _

New CECubeColumn("TVPI", "TVPI", GetType(Double)), _

New CECubeColumn("DPI", "DPI", GetType(Double)), _

New CECubeColumn("RVPI", "RVPI", GetType(Double)), _

New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)), _

New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",
GetType(Double)), _

New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _

New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _

New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _

New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _

New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _

}
```

## 6.9.6. FundInvestorPositions

### 6.9.6.1. FUNDINVESTORPOSITIONS

```
New CECubeColumn() { _
```

```
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",
GetType(DateTime)), _

New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _

New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)),
-

New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",
GetType(String)), _

New CECubeColumn("INVESTMENT_ID", "Subscriber ID{F}ID du souscripteur",
GetType(String)), _

New CECubeColumn("INVESTMENTNAME", "Subscriber name{F}Nom du
souscripteur", GetType(String)), _

New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de
l'investisseur", GetType(String)), _

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des
montants", GetType(String)), _

New CECubeColumn("INVESTMENTEXITDATE", "Exit date{F}Date de sortie",
GetType(Date)), _

New CECubeColumn("ISEXITED", "Exit?{F}Sorti ?", GetType(Boolean)), _

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",
GetType(Decimal)), _
```

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("PERCENTCALLED", "% called{F}% appelé", GetType(Double)), \_

—

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("CURRENTCOST", "Curent cost", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

```
New CECubeColumn("OTHERFEES_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), _  
New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), _  
New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), _  
New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), _  
New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), _  
New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _  
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _  
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus", GetType(Decimal)), _  
New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV", GetType(Decimal)), _  
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV", GetType(DateTime)), _  
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée", GetType(Decimal)), _  
New CECubeColumn("IRRNET", "Net IRR{F}TRI net", GetType(Double)), _  
New CECubeColumn("TVPI", "TVPI", GetType(Double)), _  
New CECubeColumn("DPI", "DPI", GetType(Double)), _  
New CECubeColumn("RVPI", "RVPI", GetType(Double)), _  
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)), _
```

```
New CECubeColumn("STAKE", "Stake %{F}% détention", GetType(Double)), _  
New CECubeColumn("STAKENOCARRIED", "Stake % (no carry){F}% détention (hors  
carry)", GetType(Double)), _  
New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",  
GetType(Double)), _  
New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _  
New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _  
New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _  
New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _  
New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _  
}  
}
```

## 6.9.7. FundSharePositions

### 6.9.7.1. FUNDSHAREPOSITIONS

```
New CECubeColumn() { _  
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",  
GetType(DateTime)), _  
New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _  
New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)),  
—  
New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",  
GetType(String)), _  
New CECubeColumn("SHARE_ID", "Share ID{F}ID part", GetType(String)), _  
New CECubeColumn("SHAREINDEX", "Share index{F}Index part", GetType(String)), _
```

New CECubeColumn("SHARECURRENCY", "Share currency{F}Devise de la part",  
GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des  
montants", GetType(String)), \_

New CECubeColumn("SHARENAME", "Share name{F}Nom de la part",  
GetType(String)), \_

New CECubeColumn("SHAREISCARRIED", "Carried ?{F}Carried ?", GetType(String)),

—

New CECubeColumn("SHAREPRICE", "Share price{F}Prix de la part", GetType(String)),

—

New CECubeColumn("COMMITTEDSHARES", "Committed shares{F}Parts engagées",  
GetType(Decimal)), \_

New CECubeColumn("AMOUNTCALLED", "Called amount{F}Montant appelé",  
GetType(Decimal)), \_

New CECubeColumn("AMOUNTDISTRIBUTED", "Distributed amount{F}Montant  
distribué", GetType(Decimal)), \_

New CECubeColumn("NBSHARES", "Nb shares{F}Nb Parts", GetType(Decimal)), \_

New CECubeColumn("SHARESCALLED", "Called shares{F}Parts appelées",  
GetType(Decimal)), \_

New CECubeColumn("SHARESBOUGHTBACK", "Bought back shares{F}Parts  
distribuées", GetType(Decimal)), \_

New CECubeColumn("SHAREVALUATION", "Share valuation{F}Valorisation de la part",  
GetType(Decimal)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",  
GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining  
commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_  
New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_  
New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_  
New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_  
New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_  
New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_  
New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_  
New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_  
New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_  
New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_  
New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_  
New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_  
New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_  
New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_  
New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

```
New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital  
rappelable", GetType(Decimal)), _  
  
New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",  
GetType(Decimal)), _  
  
New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _  
  
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _  
  
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",  
GetType(Decimal)), _  
  
New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",  
GetType(Decimal)), _  
  
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",  
GetType(DateTime)), _  
  
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",  
GetType(Decimal)) _  
  
}
```

## 6.9.8. FundInvestorSharePositions

### 6.9.8.1. FUNDINVESTORSHAREPOSITIONS

```
New CECubeColumn() { _  
  
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",  
GetType(DateTime)), _  
  
New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _  
  
New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)),  
-  
New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",  
GetType(String)), _
```

New CECubeColumn("INVESTMENT\_ID", "Subscriber ID{F}ID du souscripteur",  
GetType(String)), \_

New CECubeColumn("INVESTMENTNAME", "Subscriber name{F}Nom du  
souscripteur", GetType(String)), \_

New CECubeColumn("INVESTOR\_ID", "Investor ID{F}ID de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de  
l'investisseur", GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des  
montants", GetType(String)), \_

New CECubeColumn("INVESTMENTEXITDATE", "Exit date{F}Date de sortie",  
GetType(Date)), \_

New CECubeColumn("ISEXITED", "Exit?{F}Sorti ?", GetType(Boolean)), \_

New CECubeColumn("SHARE\_ID", "Share ID{F}ID part", GetType(String)), \_

New CECubeColumn("SHAREINDEX", "Share index{F}Index part", GetType(String)), \_

New CECubeColumn("SHARECURRENCY", "Share currency{F}Devise de la part",  
GetType(String)), \_

New CECubeColumn("SHARENAME", "Share name{F}Nom de la part",  
GetType(String)), \_

New CECubeColumn("SHAREISCARRIED", "Carried ?{F}Carried ?", GetType(String)),

-

New CECubeColumn("SHAREPRICE", "Share price{F}Prix de la part", GetType(String)), \_

New CECubeColumn("COMMITTEDSHARES", "Committed shares{F}Parts engagées", GetType(Decimal)), \_

New CECubeColumn("AMOUNTCALLED", "Called amount{F}Montant appelé", GetType(Decimal)), \_

New CECubeColumn("AMOUNTDISTRIBUTED", "Distributed amount{F}Montant distribué", GetType(Decimal)), \_

New CECubeColumn("NBSHARES", "Nb shares{F}Nb Parts", GetType(Decimal)), \_

New CECubeColumn("SHARESCALLED", "Called shares{F}Parts appelées", GetType(Decimal)), \_

New CECubeColumn("SHARESBOUGHTBACK", "Bought back shares{F}Parts distribuées", GetType(Decimal)), \_

New CECubeColumn("SHAREVALUATION", "Share valuation{F}Valorisation de la part", GetType(Decimal)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement", GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

```
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",
GetType(DateTime)), _

New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",
GetType(Decimal)), _

New CECubeColumn("IRRNET", "Net IRR{F}TRI net", GetType(Double)), _

New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)), _

New CECubeColumn("TWRINCEPTION", "TWR since inception{F}TWR depuis origine",
GetType(Double)), _

New CECubeColumn("TWR1Y", "TWR in 1 year{F}TWR sur 1 an", GetType(Double)), _

New CECubeColumn("TWR2Y", "TWR in 2 year{F}TWR sur 2 an", GetType(Double)), _

New CECubeColumn("TWR3Y", "TWR in 3 year{F}TWR sur 3 an", GetType(Double)), _

New CECubeColumn("TWR4Y", "TWR in 4 year{F}TWR sur 4 an", GetType(Double)), _

New CECubeColumn("TWR5Y", "TWR in 5 year{F}TWR sur 5 an", GetType(Double)) _

}
```

## 6.9.9. FundCashFlows

### 6.9.9.1. FUNDCASHFLOWS

```
New CECubeColumn() { _

New CECubeColumn("CASHFLOW_ID", "Cashflow ID{F}ID du cashflow",
GetType(String)), _

New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _
```

New CECubeColumn("SHARE\_ID", "Share ID{F}ID part", GetType(String)), \_

New CECubeColumn("INVESTMENT\_ID", "Subscriber ID{F}ID du souscripteur", GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des montants", GetType(String)), \_

New CECubeColumn("OPERATION\_ID", "Operation ID{F}ID opération", GetType(String)), \_

New CECubeColumn("OPERATIONTYPE", "Operation type{F}Type opération", GetType(Int32)), \_

New CECubeColumn("OPERATIONSUBTYPE", "Operation subtype{F}Sous-type opération", GetType(Int32)), \_

New CECubeColumn("OPERATIONDATE", "Operation date{F}Date opération", GetType(DateTime)), \_

New CECubeColumn("OPERATIONNAME", "Operation name{F}Nom opération", GetType(String)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement", GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_INSIDE", "Capital called inside commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALCALLED\_OUTSIDE", "Capital called outside commitment{F}Capital appelé hors engagement", GetType(Decimal)), \_

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_INSIDE", "Management fees inside commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("MANAGEMENTFEES\_OUTSIDE", "Management fees outside commitment{F}Frais de gestion hors engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

```
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
GetType(Decimal)), _

New CECubeColumn("VALUATION", "NAV{F}NAV", GetType(Decimal)), _

New CECubeColumn("VALUATIONDELTA", "Valuation delta{F}Delta valorisation",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",
GetType(DateTime)) _

}
```

## 6.9.10. InvestorPositions

### 6.9.10.1. INVESTORPOSITIONS

```
New CECubeColumn() {_

New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",
GetType(DateTime)), _

New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",
GetType(String)), _

New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de
l'investisseur", GetType(String)), _

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des
montants", GetType(String)), _
```

```
New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",
GetType(Decimal)), _

New CECubeColumn("REMAININGCOMMITMENT", "Remaining
commitment{F}Engagement résiduel", GetType(Decimal)), _

New CECubeColumn("PERCENTCALLED", "% called{F}% appelé", GetType(Double)),
-

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), _

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), _

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), _

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé",
GetType(Decimal)), _

New CECubeColumn("CAPITALCALLED_INSIDE", "Capital called inside
commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), _

New CECubeColumn("CAPITALCALLED_OUTSIDE", "Capital called outside
commitment{F}Capital appelé hors engagement", GetType(Decimal)), _

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi",
GetType(Decimal)), _

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion",
GetType(Decimal)), _

New CECubeColumn("MANAGEMENTFEES_INSIDE", "Management fees inside
commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), _

New CECubeColumn("MANAGEMENTFEES_OUTSIDE", "Management fees outside
commitment{F}Frais de gestion hors engagement", GetType(Decimal)), _

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), _

New CECubeColumn("OTHERFEES_INSIDE", "Other fees inside commitment{F}Autres
frais dans l'engagement", GetType(Decimal)), _
```

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus", GetType(Decimal)), \_

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV", GetType(Decimal)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV", GetType(DateTime)), \_

New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée", GetType(Decimal)), \_

New CECubeColumn("IRRNET", "Net IRR{F}TRI net", GetType(Double)), \_

New CECubeColumn("IRRNETREALIZED", "Realized net IRR{F}TRI net cédé", GetType(Double)), \_

New CECubeColumn("IRRNETUNREALIZED", "Unrealized net IRR{F}TRI net non cédé", GetType(Double)), \_

New CECubeColumn("TVPI", "TVPI", GetType(Double)), \_

```
New CECubeColumn("DPI", "DPI", GetType(Double)), _  
New CECubeColumn("RVPI", "RVPI", GetType(Double)), _  
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)) _  
}  
}
```

## 6.9.11. MasterfundInvestorPositions

### 6.9.11.1. MASTERFUNDINVESTORPOSITIONS

```
New CECubeColumn() { _  
New CECubeColumn("POSITION_DATE", "Position Date{F}Date position",  
GetType(DateTime)), _  
New CECubeColumn("FUND_ID", "Fund ID{F}ID du Fonds", GetType(String)), _  
New CECubeColumn("FUNDNAME", "Fund name{F}Nom du Fonds", GetType(String)),  
—  
New CECubeColumn("FUNDCURRENCY", "Fund currency{F}Devise du Fonds",  
GetType(String)), _  
New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",  
GetType(String)), _  
New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de  
l'investisseur", GetType(String)), _  
New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des  
montants", GetType(String)), _
```

```
New CECubeColumn("INVESTMENTEXITDATE", "Exit date{F}Date de sortie",
GetType(Date)), _

New CECubeColumn("ISEXITED", "Exit?{F}Sorti ?", GetType(Boolean)), _

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",
GetType(Decimal)), _

New CECubeColumn("REMAININGCOMMITMENT", "Remaining
commitment{F}Engagement résiduel", GetType(Decimal)), _

New CECubeColumn("PERCENTCALLED", "% called{F}% appelé", GetType(Double)),
-

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), _

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), _

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), _

New CECubeColumn("CAPITALCALLED", "Capital called{F}Capital appelé",
GetType(Decimal)), _

New CECubeColumn("CAPITALCALLED_INSIDE", "Capital called inside
commitment{F}Capital appelé dans l'engagement", GetType(Decimal)), _

New CECubeColumn("CAPITALCALLED_OUTSIDE", "Capital called outside
commitment{F}Capital appelé hors engagement", GetType(Decimal)), _

New CECubeColumn("CAPITALINVESTED", "Invested capital{F}Capital investi",
GetType(Decimal)), _

New CECubeColumn("MANAGEMENTFEES", "Management fees{F}Frais de gestion",
GetType(Decimal)), _

New CECubeColumn("MANAGEMENTFEES_INSIDE", "Management fees inside
commitment{F}Frais de gestion dans l'engagement", GetType(Decimal)), _

New CECubeColumn("MANAGEMENTFEES_OUTSIDE", "Management fees outside
commitment{F}Frais de gestion hors engagement", GetType(Decimal)), _
```

New CECubeColumn("OTHERFEES", "Other fees{F}Autres frais", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_INSIDE", "Other fees inside commitment{F}Autres frais dans l'engagement", GetType(Decimal)), \_

New CECubeColumn("OTHERFEES\_OUTSIDE", "Other fees outside commitment{F}Autres frais hors engagement", GetType(Decimal)), \_

New CECubeColumn("DISTRIBUTION", "Distributions{F}Distributions", GetType(Decimal)), \_

New CECubeColumn("RETURNOFCAPITAL", "Return of capital{F}Retour de capital", GetType(Decimal)), \_

New CECubeColumn("REDRAWABLECAPITAL", "Redrawable capital{F}Capital rappelable", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values", GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus", GetType(Decimal)), \_

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV", GetType(Decimal)), \_

New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV", GetType(DateTime)), \_

New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée", GetType(Decimal)), \_

New CECubeColumn("IRRNET", "Net IRR{F}TRI net", GetType(Double)), \_

New CECubeColumn("TVPI", "TVPI", GetType(Double)), \_

```
New CECubeColumn("DPI", "DPI", GetType(Double)), _  
New CECubeColumn("RVPI", "RVPI", GetType(Double)), _  
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)), _  
New CECubeColumn("STAKE", "Stake %{F}% détention", GetType(Double)) _  
} _
```

## 6.9.12. FV\_T\_FUND\_TRANSACTIONS

### 6.9.12.1. FV\_T\_FUND\_TRANSACTIONS

OPERATION\_ID;ID fund operation{F}ID opération fonds

OPERATION\_NAME;Name fund operation{F}Libellé opération fonds

FUND\_ID;ID fund{F}ID Fonds

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

REFERENCE\_DATE;Date{F}Date

INDEX;Index{F}Index

TYPE;Type{F}Type

OPERATION\_TYPE;Formatted type{F}Type formatté

TRANSACTION\_CANCELLED;Cancelled

DRAFT;Draft{F}Brouillon

IRR\_DATE;Date for IRR

EXPL\_STAKE;Explicit stake{F}% détention explicite

EXPL\_STAKE\_NC;Explicit stake no carried{F}% détention explicite hors carried

VALUATION;Valuation{F}Valorisation

VALUATION2;Valuation(curr2)

COST;Cost{F}Prix de revient

COST2;Cost(curr2)

INVESTMENT\_ID;ID investment{F}ID Investissement

COMMITMENT;Commitment{F}Engagement

COMMITMENT1;Commitment(1){F}Engagement(1)

COMMITMENT2;Commitment(2){F}Engagement(2)

REMAINING\_COMMITMENT;Remaining commitment

REMAINING\_COMMITMENT1;Remaining commitment(1)

REMAINING\_COMMITMENT2;Remaining commitment(2)

CASH;Cash amount{F}Montant cash

CASH1;Cash amount(1){F}Montant cash(1)

CASH2;Cash amount(2){F}Montant cash(2)

CAPITAL\_CALL;Capital call{F}Capital appelé

CAPITAL\_CALL1;Capital call(1){F}Capital appelé(1)

CAPITAL\_CALL2;Capital call(2){F}Capital appelé(2)

CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Capital appelé hors engagement

CAPITAL\_CALL\_OUTSIDE\_COMM1;Capital call outside commitment(1){F}Capital appelé hors engagement(1)

CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(2){F}Capital appelé hors engagement(2)

DISTRIBUTION;Distribution{F}Distribution

DISTRIBUTION1;Distribution(1){F}Distribution(1)

DISTRIBUTION2;Distribution(2){F}Distribution(2)

MANAGEMENT\_FEES;Management fees{F}Frais de gestion

MANAGEMENT\_FEES1;Management fees(1){F}Frais de gestion(1)

MANAGEMENT\_FEES2;Management fees(2){F}Frais de gestion(2)

MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement

MANAGEMENT\_FEES\_OUTSIDE\_COMM1;Management fees outside commitment(1){F}Frais de gestion hors engagement(1)

MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(2){F}Frais de gestion hors engagement(2)

OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES1;Other fees(1){F}Autres frais(1)

OTHER\_FEES2;Other fees(2){F}Autres frais(2)

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement

OTHER\_FEES\_OUTSIDE\_COMM1;Other fees outside commitment(1){F}Autres frais hors engagement(1)

OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(2){F}Autres frais hors engagement(2)

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé

RETURN\_OF\_CAPITAL1;Return of capital(1){F}Capital remboursé(1)

RETURN\_OF\_CAPITAL2;Return of capital(2){F}Capital remboursé(2)

REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

REDRAWABLE\_CAPITAL1;Redrawable call(1){F}Distribution temporaire(1)

REDRAWABLE\_CAPITAL2;Redrawable call(2){F}Distribution temporaire(2)

CARRY;Carry{F}Intérêt prioritaire

CARRY1;Carry(1){F}Intérêt prioritaire(1)

CARRY2;Carry(2){F}Intérêt prioritaire(2)

GAIN\_LOSS;Gain/Loss{F}Plus/moins value

GAIN\_LOSS1;Gain/Loss(1){F}Plus/moins value(1)

GAIN\_LOSS2;Gain/Loss(2){F}Plus/moins value(2)

INTERESTS;Interests{F}Intérêts

INTERESTS1;Interests(1){F}Intérêts(1)

INTERESTS2;Interests(2){F}Intérêts(2)

DIVIDENDS;Dividends{F}Dividendes

DIVIDENDS1;Dividends(1){F}Dividendes(1)

DIVIDENDS2;Dividends(2){F}Dividendes(2)

OTHER\_INCOME;Other income{F}Autres revenus

OTHER\_INCOME1;Other income(1){F}Autres revenus(1)

OTHER\_INCOME2;Other income(2){F}Autres revenus(2)

TRANSACTION\_COSTS;Transaction costs{F}Coûts de transaction

TRANSACTION\_COSTS1;Transaction costs(1){F}Coûts de transaction(1)

TRANSACTION\_COSTS2;Transaction costs(2){F}Coûts de transaction(2)

COMPANY\_ID;Company ID{F}ID société

COMPANY\_NAME;Company name{F}Société

DIK\_SHARES;Number of shares{F}Nombre de titres

DIK\_VALUE\_PER\_SHARE;Value per share{F}Valeur du titre

DIK\_COST\_PER\_SHARE;Cost per share{F}Prix de revient du titre

SOS\_CASH;SOS Cash amount{F}Montant cession titres

SOS\_CASH1;SOS Cash amount(1){F}Montant cession titres(1)

SOS\_CASH2;SOS Cash amount(2){F}Montant cession titres(2)

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

SHAREINDEX;

VALUATION1;

COST1;

SECURITY\_CURRENCY;Share currency{F}Devise part

FUND\_NAME;Fund name{F}Nom du fonds

FUND\_CURRENCY;Fund currency{F}Devise fonds

INIT\_COMMITMENT;Initial commitment{F}Engagement initial

INVESTMENT\_NAME;Investment name{F}Nom investissement

INVESTMENT\_TYPE\_CODE;Type

INVESTMENT\_TYPE;Type

INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement

INVESTOR\_FUND\_ID;ID fund investor{F}ID fonds Investisseur

INVESTOR\_FUND\_NAME;Name fund investor{F}Nom fonds Investisseur

INVESTOR\_FUND\_CURRENCY;Currency fund investor{F}Devise fonds Investisseur

INVESTOR\_FUND\_STATUS;Status fund investor{F}Statut fonds Investisseur

INVESTOR\_COMPANY\_ID;ID company investor{F}ID société Investisseur

INVESTOR\_COMPANY\_NAME;Name company investor{F}Nom société Investisseur

INVESTOR\_COMPANY\_CURRENCY;Currency company investor{F}Devise société Investisseur

INVESTOR\_COMPANY\_STATUS;Status company investor{F}Statut société Investisseur

INVESTOR\_CONTACT\_ID;ID contact investor{F}ID contact Investisseur

INVESTOR\_CONTACT\_LASTNAME;Lastname contact investor{F}Nom contact Investisseur

INVESTOR\_CONTACT\_FIRSTNAME;Firstname contact investor{F}Prénom contact Investisseur

INVESTOR\_CONTACT\_CURRENCY;Currency contact investor{F}Devise contact Investisseur

INVESTOR\_CONTACT\_STATUS;Status contact investor{F}Statut contact Investisseur

ID\_CURRTABLE\_FUND;

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

RATE;FX rate (Investor / Share){F}Taux de change (Investisseur / Part)

RATE1;FX rate (Fund / Share){F}Taux de change (Fonds / Part)

CASH\_OUT;Cash out

CASH\_OUT1;Cash out(1)

CASH\_OUT2;Cash out(2)

CASH\_IN;Cash in

CASH\_IN1;Cash in(1)

CASH\_IN2;Cash in(2)

ADJUSTED\_VALUATION;Adjusted valuation{F}Valorisation ajustée

ADJUSTED\_VALUATION1;Adjusted valuation(1){F}Valorisation ajustée(1)

ADJUSTED\_VALUATION2;Adjusted valuation(2){F}Valorisation ajustée(2)

LAST\_VALUATION\_DATE;Last valuation date{F}Date dernière valorisation

SECURITY\_COMMITMENT;Nb shares committed{F}Nb parts engagées

SECURITY\_NOMINAL;Nominal ajusment{F}Ajustement nominal  
SECURITY\_NOMINAL1;Nominal ajusment(1){F}Ajustement nominal(1)  
SECURITY\_NOMINAL2;Nominal ajusment(2){F}Ajustement nominal(2)  
SECURITY\_NOMINAL\_CALLED;Nominal called{F}Nominal appelé  
SECURITY\_NOMINAL\_CALLED1;Nominal called(1){F}Nominal appelé(1)  
SECURITY\_NOMINAL\_CALLED2;Nominal called(2){F}Nominal appelé(2)  
SECURITY\_NOMINAL\_BOUGHTBACK;Nominal bought back{F}Nominal racheté  
SECURITY\_NOMINAL\_BOUGHTBACK1;Nominal bought back(1){F}Nominal racheté(1)  
SECURITY\_NOMINAL\_BOUGHTBACK2;Nominal bought back(2){F}Nominal racheté(2)  
SECURITY\_VALUE;Security value{F}Valeur part  
SECURITY\_VALUE1;Security value(1){F}Valeur part(1)  
SECURITY\_VALUE2;Security value(2){F}Valeur part(2)  
SECURITY\_CALLED;Security called{F}Appelé part  
SECURITY\_CALLED1;Security called(1){F}Appelé part(1)  
SECURITY\_CALLED2;Security called(2){F}Appelé part(2)  
SECURITY\_DISTRIBUTED;Security distributed{F}Distribué part  
SECURITY\_DISTRIBUTED1;Security distributed(1){F}Distribué part(1)  
SECURITY\_DISTRIBUTED2;Security distributed(2){F}Distribué part(2)  
NB SHARES;Nb shares{F}Nb parts  
NB SHARES\_CALLED;Nb shares called{F}Nb parts appelées  
NB SHARES\_BOUGHTBACK;Nb shares bought back{F}Nb parts rachetées

SHARE\_ID;Share ID{F}ID Part

SHARE\_TICKER;Share Ticker code{F}Code Ticker de la part

SHARE\_ISIN;Share ISIN code{F}Code ISIN de la part

SHARE\_SICOVAM;Share SICOVAM code{F}Code SICOVAM de la part

USE\_SECONDARY\_CURRENCY;Subscription currency is commitment currency

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Capital appelé dans l'engagement

CAPITAL\_CALL\_INSIDE\_COMM1;Capital call inside commitment(1){F}Capital appelé dans l'engagement(1)

CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Capital appelé dans l'engagement(2)

SECURITY\_FUND\_ID;ID Fund share{F}ID Part

SECURITY\_ISCARRIED;Carried share{F}Part de carried

SECURITY\_NAME;Shortname{F}Abbréviation

SECURITY\_COMPLETE\_NAME;Name{F}Nom

SECURITY\_PRICE1;Price{F}Nominal

NBSHARE\_ROUNDING;Number of shares rounding{F}Arrondi du nombre de parts

NBSHARE\_ROUNDING\_TYPE;Number of share rounding mode

SHAREPRICE\_ROUNDING;Share price rounding{F}Arrondi du nominal

SHAREPRICE\_ROUNDING\_TYPE;Share price rounding mode{F}Type d'arrondi par part

IS\_ROUNDINGSHARE\_CLASS;Is rounding share class{F}Classe réceptionnant les arrondis

COMMITMENT\_IN\_AMOUNT;Commitment in amount

SECURITY\_ID;Security ID{F}ID part

SECURITY\_PRICE;Price{F}Nominal

SECURITY\_PRICE2;Price(2){F}Nominal(2)

NB SHARES\_ROUNDED;Nb shares rounded{F}Nb parts arrondi

NB SHARES\_CALLED\_ROUNDED;Nb shares called rounded{F}Nb parts appelées arrondi

NB SHARES\_BOUGHTBACK\_ROUNDED;Nb shares bought back rounded{F}Nb parts rachetées arrondi

YEAR;Year{F}Année

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

## 6.9.13. FV\_T\_FUND\_SEC\_POSITIONS\_BY\_INVESTMENTS

### 6.9.13.1. FV\_T\_FUND\_SEC\_POSITIONS\_BY\_INVESTMENTS

INVESTMENT\_ID;ID investment{F}ID Investissement

SHAREINDEX;

FUND\_ID;ID fund{F}ID Fonds

FUND\_NAME;Fund name{F}Nom du fonds

FUND\_CURRENCY;Fund currency{F}Devise fonds

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

INVESTMENT\_NAME;Investment name{F}Nom investissement

INVESTMENT\_TYPE\_CODE;Type

INVESTMENT\_TYPE;Type

INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

SHARE\_ID;Share ID{F}ID Part

SECURITY\_FUND\_ID;ID Fund share{F}ID Part

SECURITY\_ID;Security ID{F}ID part

SECURITY\_CURRENCY;Share currency{F}Devise part

SECURITY\_PRICE1;Price{F}Nominal

SECURITY\_ISCARRIED;Carried share{F}Part de carried

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

ID\_CURRTABLE\_FUND;

NBSHARE\_ROUNDING;Number of shares rounding{F}Arrondi du nombre de parts

NBSHARE\_ROUNDING\_TYPE;Number of share rounding mode

SHAREPRICE\_ROUNDING;Share price rounding{F}Arrondi du nominal

SHAREPRICE\_ROUNDING\_TYPE;Share price rounding mode{F}Type d'arrondi par part

IS\_ROUNDINGSHARE\_CLASS;Is rounding share class{F}Classe réceptionnant les arrondis

COMMITMENT\_IN\_AMOUNT;Commitment in amount

RATE\_INV\_TO\_SEC;

RATE\_SEC\_TO\_INV;

RATE\_SEC\_TO\_FUND;

SECURITY\_NB SHARES\_COMMITTED;Nb shares commited{F}Nb parts engagées

SECURITY\_CALLED;Security called{F}Appelé part

SECURITY\_CALLED1;Security called(1){F}Appelé part(1)

SECURITY\_CALLED2;Security called(2){F}Appelé part(2)

SECURITY\_DISTRIBUTED;Security distributed{F}Distribué part

SECURITY\_DISTRIBUTED1;Security distributed(1){F}Distribué part(1)

SECURITY\_DISTRIBUTED2;Security distributed(2){F}Distribué part(2)

NB\_SHARES;Nb shares{F}Nombre de parts

NB\_SHARES\_CALLED;Nb shares called{F}Nombre de parts appelées

NB\_SHARES\_BOUGHTBACK;Nb shares bought back{F}Nombre de parts rachetées

REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel

REMAINING\_COMMITMENT1;Remaining commitment(1){F}Engagement résiduel(1)

REMAINING\_COMMITMENT2;Remaining commitment(2){F}Engagement résiduel(2)

CAPITAL\_CALL;Capital call{F}Montant appelé

CAPITAL\_CALL1;Capital call(1){F}Montant appelé(1)

CAPITAL\_CALL2;Capital call(2){F}Montant appelé(2)

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement

CAPITAL\_CALL\_INSIDE\_COMM1;Capital call inside commitment(1){F}Montant appelé dans l'engagement(1)

CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)

CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement

CAPITAL\_CALL\_OUTSIDE\_COMM1;Capital call outside commitment(1){F}Montant appelé hors engagement(1)

CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(2){F}Montant appelé hors engagement(2)

MANAGEMENT\_FEES;Management fees{F}Frais de gestion

MANAGEMENT\_FEES1;Management fees(1){F}Frais de gestion(1)

MANAGEMENT\_FEES2;Management fees(2){F}Frais de gestion(2)

MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement

MANAGEMENT\_FEES\_OUTSIDE\_COMM1;Management fees outside commitment(1){F}Frais de gestion hors engagement(1)

MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(2){F}Frais de gestion hors engagement(2)

OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES1;Other fees(1){F}Autres frais(1)

OTHER\_FEES2;Other fees(2){F}Autres frais(2)

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement

OTHER\_FEES\_OUTSIDE\_COMM1;Other fees outside commitment(1){F}Autres frais hors engagement(1)

OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(2){F}Autres frais hors engagement(2)

DISTRIBUTION;Distributions{F}Distributions

DISTRIBUTION1;Distributions(1){F}Distributions(1)

DISTRIBUTION2;Distributions(2){F}Distributions(2)

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé

RETURN\_OF\_CAPITAL1;Return of capital(1){F}Capital remboursé(1)

RETURN\_OF\_CAPITAL2;Return of capital(2){F}Capital remboursé(2)

REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

REDRAWABLE\_CAPITAL1;Redrawable call(1){F}Distribution temporaire(1)

REDRAWABLE\_CAPITAL2;Redrawable call(2){F}Distribution temporaire(2)

CARRY;Carry{F}Intérêt prioritaire

CARRY1;Carry(1){F}Intérêt prioritaire(1)

CARRY2;Carry(2){F}Intérêt prioritaire(2)

GAIN\_LOSS;Gain/Loss{F}Plus/moins value

GAIN\_LOSS1;Gain/Loss(1){F}Plus/moins value(1)

GAIN\_LOSS2;Gain/Loss(2){F}Plus/moins value(2)

INTERESTS;Interests{F}Intérêts

INTERESTS1;Interests(1){F}Intérêts(1)

INTERESTS2;Interests(2){F}Intérêts(2)

DIVIDENDS;Dividends{F}Dividendes

DIVIDENDS1;Dividends(1){F}Dividendes(1)

DIVIDENDS2;Dividends(2){F}Dividendes(2)

OTHER\_INCOME;Other income{F}Autres revenus

OTHER\_INCOME1;Other income(1){F}Autres revenus(1)

OTHER\_INCOME2;Other income(2){F}Autres revenus(2)

TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions

TRANSACTION\_COSTS1;Transaction costs(1){F}Coûts des transactions(1)

TRANSACTION\_COSTS2;Transaction costs(2){F}Coûts des transactions(2)

COST;Cost{F}Prix de revient

COST1;Cost(1){F}Prix de revient(1)

COST2;Cost(2){F}Prix de revient(2)

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

SECURITY\_NAME;Security name{F}Nom part

SECURITY\_VALUE1;Security value(1){F}Valeur part(1)

LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)

LAST\_FUNDAMIN\_NOMINAL\_VALUATION1;Last valuation (fund admin)(1){F}Dernière valorisation (admin de fonds)(1)

LAST\_FUNDAMIN\_NB SHARES;Nb shares for last valuation(fund admin){F}Nombre de parts à la dernière valo(admin de fonds)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION1;Last rounded total valuation (fund admin)(1){F}Dernière valorisation totale arrondie (admin de fonds)(1)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION;Last rounded total valuation (fund admin){F}Dernière valorisation totale arrondie (admin de fonds)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION2;Last rounded total valuation (fund admin)(2){F}Dernière valorisation totale arrondie (admin de fonds)(2)

SECURITY\_NOMINAL\_CALLED;Nominal called{F}Nominal appelé

SECURITY\_NOMINAL\_CALLED1;Nominal called(1){F}Nominal appelé(1)

SECURITY\_NOMINAL\_BOUGHTBACK;Nominal bought{F}Nominal racheté

SECURITY\_NOMINAL\_BOUGHTBACK1;Nominal bought(1){F}Nominal racheté(1)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION1;Last rounded nominal valuation (fund admin)(1){F}Dernière valorisation nominale arrondie (admin de fonds)(1)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION;Last rounded nominal valuation (fund admin){F}Dernière valorisation nominale arrondie (admin de fonds)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION2;Last rounded nominal valuation (fund admin)(2){F}Dernière valorisation nominale arrondie (admin de fonds)(2)

COMMITMENT;Total commitment{F}Engagement total

COMMITMENT1;Total commitment(1){F}Engagement total(1)

COMMITMENT2;Total commitment(2){F}Engagement total(2)

SECURITY\_PRICE;Price{F}Nominal

SECURITY\_PRICE2;Price(2){F}Nominal(2)

SECURITY\_VALUE;Security value{F}Valeur part

SECURITY\_VALUE2;Security value(2){F}Valeur part(2)

SECURITY\_TOTAL\_COMMITMENT;Share total commitment{F}Engagement total part

SECURITY\_TOTAL\_COMMITMENT1;Share total commitment(1){F}Engagement total part(1)

SECURITY\_TOTAL\_COMMITMENT2;Share total commitment(2){F}Engagement total part(2)

SECURITY\_TOTAL\_VALUE;Share total value{F}Valeur totale part

SECURITY\_TOTAL\_VALUE1;Share total value(1){F}Valeur totale part(1)

SECURITY\_TOTAL\_VALUE2;Share total value(2){F}Valeur totale part(2)

SECURITY\_NOMINAL;Current nominal{F}Nominal courant

SECURITY\_NOMINAL1;Current nominal(1){F}Nominal courant(1)

SECURITY\_NOMINAL2;Current nominal(2){F}Nominal courant(2)

LAST\_FUNDAMIN\_NOMINAL\_VALUATION;Last valuation (fund admin){F}Dernière valorisation (admin de fonds)

LAST\_FUNDADMIN\_NOMINAL\_VALUATION2;Last valuation (fund admin)(2){F}Dernière valorisation (admin de fonds)(2)

SECURITY\_NB SHARES COMMITTED\_ROUNDED;Committed securities number{F}Nb parts engagées

NB SHARES\_ROUNDED;Nb shares{F}Nombre de parts

NB SHARES\_CALLED\_ROUNDED;Nb shares called{F}Nombre de parts appelées

NB SHARES\_BOUGHTBACK\_ROUNDED;Nb shares bought back{F}Nombre de parts rachetées

NET\_IRR\_BY\_SECURITY;Net IRR by Investor security{F}TRI NET par part investisseur

LAST\_NAV\_DATE;Date{F}Date(first)

## 6.9.14. FV\_T\_FUND\_SEC\_POSITIONS\_BY\_FUNDS

### 6.9.14.1. FV\_T\_FUND\_SEC\_POSITIONS\_BY\_FUNDS

INVESTMENT\_ID;ID investment{F}ID Investissement

SHAREINDEX;

FUND\_ID;ID fund{F}ID Fonds

FUND\_NAME;Fund name{F}Nom du fonds

FUND\_CURRENCY;Fund currency{F}Devise fonds

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

INVESTMENT\_NAME;Investment name{F}Nom investissement

INVESTMENT\_TYPE\_CODE;Type

INVESTMENT\_TYPE;Type

INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

SHARE\_ID;Share ID{F}ID Part

SECURITY\_FUND\_ID;ID Fund share{F}ID Part

SECURITY\_ID;Security ID{F}ID part

SECURITY\_CURRENCY;Share currency{F}Devise part

SECURITY\_PRICE1;Price{F}Nominal

SECURITY\_ISCARRIED;Carried share{F}Part de carried

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

ID\_CURRTABLE\_FUND;

NBSHARE\_ROUNDING;Number of shares rounding{F}Arrondi du nombre de parts

NBSHARE\_ROUNDING\_TYPE;Number of share rounding mode

SHAREPRICE\_ROUNDING;Share price rounding{F}Arrondi du nominal

SHAREPRICE\_ROUNDING\_TYPE;Share price rounding mode{F}Type d'arrondi par part

IS\_ROUNDINGSHARE\_CLASS;Is rounding share class{F}Classe réceptionnant les arrondis

COMMITMENT\_IN\_AMOUNT;Commitment in amount

RATE\_INV\_TO\_SEC;

RATE\_SEC\_TO\_INV;

RATE\_SEC\_TO\_FUND;

SECURITY\_NB\_SHARES\_COMMITTED;Nb shares commited{F}Nb parts engagées

SECURITY\_CALLED;Security called{F}Appelé part

SECURITY\_CALLED1;Security called(1){F}Appelé part(1)

SECURITY\_CALLED2;Security called(2){F}Appelé part(2)

SECURITY\_DISTRIBUTED;Security distributed{F}Distribué part

SECURITY\_DISTRIBUTED1;Security distributed(1){F}Distribué part(1)

SECURITY\_DISTRIBUTED2;Security distributed(2){F}Distribué part(2)

NB\_SHARES;Nb shares{F}Nombre de parts

NB\_SHARES\_CALLED;Nb shares called{F}Nombre de parts appelées

NB\_SHARES\_BOUGHTBACK;Nb shares bought back{F}Nombre de parts rachetées

REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel

REMAINING\_COMMITMENT1;Remaining commitment(1){F}Engagement résiduel(1)

REMAINING\_COMMITMENT2;Remaining commitment(2){F}Engagement résiduel(2)

CAPITAL\_CALL;Capital call{F}Montant appelé

CAPITAL\_CALL1;Capital call(1){F}Montant appelé(1)

CAPITAL\_CALL2;Capital call(2){F}Montant appelé(2)

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement

CAPITAL\_CALL\_INSIDE\_COMM1;Capital call inside commitment(1){F}Montant appelé dans l'engagement(1)

CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)

CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement

CAPITAL\_CALL\_OUTSIDE\_COMM1;Capital call outside commitment(1){F}Montant appelé hors engagement(1)

CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(2){F}Montant appelé hors engagement(2)

MANAGEMENT\_FEES;Management fees{F}Frais de gestion

MANAGEMENT\_FEES1;Management fees(1){F}Frais de gestion(1)

MANAGEMENT\_FEES2;Management fees(2){F}Frais de gestion(2)

MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement

MANAGEMENT\_FEES\_OUTSIDE\_COMM1;Management fees outside commitment(1){F}Frais de gestion hors engagement(1)

MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(2){F}Frais de gestion hors engagement(2)

OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES1;Other fees(1){F}Autres frais(1)

OTHER\_FEES2;Other fees(2){F}Autres frais(2)

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement

OTHER\_FEES\_OUTSIDE\_COMM1;Other fees outside commitment(1){F}Autres frais hors engagement(1)

OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(2){F}Autres frais hors engagement(2)

DISTRIBUTION;Distributions{F}Distributions

DISTRIBUTION1;Distributions(1){F}Distributions(1)

DISTRIBUTION2;Distributions(2){F}Distributions(2)

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé

RETURN\_OF\_CAPITAL1;Return of capital(1){F}Capital remboursé(1)

RETURN\_OF\_CAPITAL2;Return of capital(2){F}Capital remboursé(2)

REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

REDRAWABLE\_CAPITAL1;Redrawable call(1){F}Distribution temporaire(1)

REDRAWABLE\_CAPITAL2;Redrawable call(2){F}Distribution temporaire(2)

CARRY;Carry{F}Intérêt prioritaire

CARRY1;Carry(1){F}Intérêt prioritaire(1)

CARRY2;Carry(2){F}Intérêt prioritaire(2)

GAIN\_LOSS;Gain/Loss{F}Plus/moins value

GAIN\_LOSS1;Gain/Loss(1){F}Plus/moins value(1)

GAIN\_LOSS2;Gain/Loss(2){F}Plus/moins value(2)

INTERESTS;Interests{F}Intérêts

INTERESTS1;Interests(1){F}Intérêts(1)

INTERESTS2;Interests(2){F}Intérêts(2)

DIVIDENDS;Dividends{F}Dividendes

DIVIDENDS1;Dividends(1){F}Dividendes(1)

DIVIDENDS2;Dividends(2){F}Dividendes(2)

OTHER\_INCOME;Other income{F}Autres revenus

OTHER\_INCOME1;Other income(1){F}Autres revenus(1)

OTHER\_INCOME2;Other income(2){F}Autres revenus(2)

TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions

TRANSACTION\_COSTS1;Transaction costs(1){F}Coûts des transactions(1)

TRANSACTION\_COSTS2;Transaction costs(2){F}Coûts des transactions(2)

COST;Cost{F}Prix de revient

COST1;Cost(1){F}Prix de revient(1)

COST2;Cost(2){F}Prix de revient(2)

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

SECURITY\_NAME;Security name{F}Nom part

SECURITY\_VALUE1;Security value(1){F}Valeur part(1)

LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)

LAST\_FUNDAMIN\_NOMINAL\_VALUATION1;Last valuation (fund admin)(1){F}Dernière valorisation (admin de fonds)(1)

LAST\_FUNDAMIN\_NB SHARES;Nb shares for last valuation(fund admin){F}Nombre de parts à la dernière valo(admin de fonds)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION1;Last rounded total valuation (fund admin)(1){F}Dernière valorisation totale arrondie (admin de fonds)(1)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION;Last rounded total valuation (fund admin){F}Dernière valorisation totale arrondie (admin de fonds)

LAST\_FUNDAMIN\_ROUNDED\_VALUATION2;Last rounded total valuation (fund admin)(2){F}Dernière valorisation totale arrondie (admin de fonds)(2)

SECURITY\_NOMINAL\_CALLED;Nominal called{F}Nominal appelé

SECURITY\_NOMINAL\_CALLED1;Nominal called(1){F}Nominal appelé(1)

SECURITY\_NOMINAL\_BOUGHTBACK;Nominal bought{F}Nominal racheté

SECURITY\_NOMINAL\_BOUGHTBACK1;Nominal bought(1){F}Nominal racheté(1)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION1;Last rounded nominal valuation (fund admin)(1){F}Dernière valorisation nominale arrondie (admin de fonds)(1)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION;Last rounded nominal valuation (fund admin){F}Dernière valorisation nominale arrondie (admin de fonds)

LAST\_FUNDAMIN\_NOMINAL\_ROUNDED\_VALUATION2;Last rounded nominal valuation (fund admin)(2){F}Dernière valorisation nominale arrondie (admin de fonds)(2)

COMMITMENT;Total commitment{F}Engagement total

COMMITMENT1;Total commitment(1){F}Engagement total(1)

COMMITMENT2;Total commitment(2){F}Engagement total(2)

SECURITY\_PRICE;Price{F}Nominal

SECURITY\_PRICE2;Price(2){F}Nominal(2)

SECURITY\_VALUE;Security value{F}Valeur part

SECURITY\_VALUE2;Security value(2){F}Valeur part(2)

SECURITY\_TOTAL\_COMMITMENT;Share total commitment{F}Engagement total part

SECURITY\_TOTAL\_COMMITMENT1;Share total commitment(1){F}Engagement total part(1)

SECURITY\_TOTAL\_COMMITMENT2;Share total commitment(2){F}Engagement total part(2)

SECURITY\_TOTAL\_VALUE;Share total value{F}Valeur totale part

SECURITY\_TOTAL\_VALUE1;Share total value(1){F}Valeur totale part(1)

SECURITY\_TOTAL\_VALUE2;Share total value(2){F}Valeur totale part(2)

SECURITY\_NOMINAL;Current nominal{F}Nominal courant

SECURITY\_NOMINAL1;Current nominal(1){F}Nominal courant(1)

SECURITY\_NOMINAL2;Current nominal(2){F}Nominal courant(2)

LAST\_FUNDAMIN\_NOMINAL\_VALUATION;Last valuation (fund admin){F}Dernière valorisation (admin de fonds)

LAST\_FUNDAMIN\_NOMINAL\_VALUATION2;Last valuation (fund admin)(2){F}Dernière valorisation (admin de fonds)(2)

SECURITY\_NB SHARES\_COMMITTED\_ROUNDED;Committed securities number{F}Nb parts engagées

NB SHARES\_ROUNDED;Nb shares{F}Nombre de parts

NB SHARES CALLED ROUNDED;Nb shares called{F}Nombre de parts appelées

NB SHARES BOUGHTBACK ROUNDED;Nb shares bought back{F}Nombre de parts rachetées

NET IRR BY SECURITY;Net IRR by Investor security{F}TRI NET par part investisseur

LAST NAV DATE;Date{F}Date(first)

## 6.9.15. FV\_T\_FUND\_POSITIONS\_BY\_FUNDS

### 6.9.15.1. FV\_T\_FUND\_POSITIONS\_BY\_FUNDS

FUND\_ID;ID fund{F}ID Fonds

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel

CASH\_IN;Cash in{F}Cash in

CASH\_OUT;Cash out{F}Cash out

CAPITAL\_CALL;Capital call{F}Montant appelé

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement

CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement

MANAGEMENT\_FEES;Management fees{F}Frais de gestion

MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement

OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement

DISTRIBUTION;Distributions{F}Distributions

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé

REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

CARRY;Carry{F}Intérêt prioritaire

GAIN\_LOSS;Gain/Loss{F}Plus/moins value

INTERESTS;Interests{F}Intérêts

DIVIDENDS;Dividends{F}Dividendes

OTHER\_INCOME;Other income{F}Autres revenus

TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions

COST;Cost{F}Prix de revient

ADJUSTED\_NAV;Adjusted NAV{F}NAV ajustée

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenues hors engagement

GAIN LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

LAST\_NAV;Last NAV{F}Dernière NAV

LAST\_NAV\_DATE;Last NAV date{F}Dernière date NAV

COMMITMENT;Total commitment{F}Engagement total

CASH\_ON\_CASH;Cash on cash{F}Cash on cash

CASH\_BALANCE;Cash balance{F}Balance CASH

NAV\_ADJUSTMENT;NAV adjustment{F}Ajustement NAV

PERCENT\_CALLED;% called{F}% appelé

TVPI;TVPI{F}TVPI

DPI;DPI{F}DPI

RVPI;RVPI{F}RVPI

NET\_IRR;Net IRR{F}TRI NET

FUND\_NAME;Fund{F}Fonds

FUND\_ABBREVIATION;Short name{F}Abréviation

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

FUND\_STATUS;Status{F}Statut

FUND\_STATUS\_FORMATTED;Status

FUND\_TYPE;Legal form

FUND\_REGION;Geography

MANAGEMENT\_COMPANY\_ID;IQId

MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion

VINTAGE\_YEAR;Vintage year{F}Millésime

FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing

FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing

CURRENCY;Currency{F}Devise

DOMICILE;Domicile{F}Domiciliation

LEGAL\_FORM;Legal form{F}Forme juridique

FUND\_SIZE;Size{F}Taille

TERM\_YEARS;Term (Years){F}Durée (années)

INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)

TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable

END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement

INVESTMENT\_PERIOD\_ADD;Extension

EVCA\_STRUCTURE;Structure

EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage

EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography

FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV

LAST\_FUNDADMIN\_VALUATION;Last total valuation (fund admin){F}Dernière valorisation totale (admin de fonds)

LAST\_FUNDADMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)

NET\_IRR\_WO\_CARRIED;Net IRR wo Carried{F}TRI NET Hors Carried

## 6.9.16. FV\_T\_FUND\_POSITIONS\_BY\_INVESTMENTS

### 6.9.16.1. FV\_T\_FUND\_POSITIONS\_BY\_FUNDS

FUND\_CURRENCY;Fund currency{F}Devise fonds

INIT\_COMMITMENT;Initial commitment{F}Engagement initial

SHAREINIT1;Share init 1{F}Titre init 1

SHAREINIT2;Share init 2{F}Titre init 2

SHAREINIT3;Share init 3{F}Titre init 3

SHAREINIT4;Share init 4{F}Titre init 4

SHAREINIT5;Share init 5{F}Titre init 5

SHAREINIT6;Share init 6{F}Titre init 6

SHAREINIT7;Share init 7{F}Titre init 7

SHAREINIT8;Share init 8{F}Titre init 8

SHAREINIT9;Share init 9{F}Titre init 9

SHAREINIT10;Share init 10{F}Titre init 10

SHAREINIT11;Share init 11{F}Titre init 11

SHAREINIT12;Share init 12{F}Titre init 12

SHAREINIT13;Share init 13{F}Titre init 13

SHAREINIT14;Share init 14{F}Titre init 14

SHAREINIT15;Share init 15{F}Titre init 15

SHAREINIT16;Share init 16{F}Titre init 16

SHAREINIT17;Share init 17{F}Titre init 17

SHAREINIT18;Share init 18{F}Titre init 18

SHAREINIT19;Share init 19{F}Titre init 19

SHAREINIT20;Share init 20{F}Titre init 20

INVESTMENT\_ID;ID investment{F}ID investissement

INVESTMENT\_NAME;Investment name{F}Nom investissement

INVESTMENT\_TYPE\_CODE;Type

INVESTMENT\_TYPE;Type

INVESTMENT\_CLOSING\_DATE;Investment closing date{F}Date de premier closing

INVESTMENT\_EXIT\_DATE;Investment exit date{F}Date de sortie investissement

INVESTMENT\_EXPL\_STAKE;Investment forced stake{F}% détention forcé

INVESTMENT\_EXPL\_STAKE\_NC;Investment forced stake NC{F}% détention forcé NC

INVESTOR\_FUND\_ID;ID fund investor{F}ID fonds Investisseur

INVESTOR\_FUND\_NAME;Name fund investor{F}Nom fonds Investisseur

INVESTOR\_FUND\_CURRENCY;Currency fund investor{F}Devise fonds Investisseur

INVESTOR\_FUND\_STATUS;Status fund investor{F}Statut fonds Investisseur

INVESTOR\_COMPANY\_ID;ID company investor{F}ID société Investisseur

INVESTOR\_COMPANY\_NAME;Name company investor{F}Nom société Investisseur

INVESTOR\_COMPANY\_CURRENCY;Currency company investor{F}Devise société  
Investisseur

INVESTOR\_COMPANY\_STATUS;Status company investor{F}Statut société Investisseur

INVESTOR\_CONTACT\_ID;ID contact investor{F}ID contact Investisseur

INVESTOR\_CONTACT\_LASTNAME;Lastname contact investor{F}Nom contact Investisseur

INVESTOR\_CONTACT\_FIRSTNAME;Firstname contact investor{F}Prénom contact Investisseur

INVESTOR\_CONTACT\_CURRENCY;Currency contact investor{F}Devise contact Investisseur

INVESTOR\_CONTACT\_STATUS;Status contact investor{F}Statut contact Investisseur

INVESTMENT\_CURRENCY;Currency(2)

FUND\_ID;ID fund{F}ID Fonds

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

COMMITMENT;Total commitment{F}Engagement total

FUND\_NAME;Fund{F}Fonds

FUND\_ABBREVIATION;Short name{F}Abréviation

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

FUND\_STATUS;Status{F}Statut

FUND\_STATUS\_FORMATTED;Status

FUND\_TYPE;Legal form

FUND\_REGION;Geography

MANAGEMENT\_COMPANY\_ID;IQId

MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion

VINTAGE\_YEAR;Vintage year{F}Millésime

FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing

FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing

CURRENCY;Currency{F}Devise

DOMICILE;Domicile{F}Domiciliation

LEGAL\_FORM;Legal form{F}Forme juridique

TOTAL\_FUND\_COMMITMENT;Size{F}Taille

TERM\_YEARS;Term (Years){F}Durée (années)

INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)

TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable

END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement

INVESTMENT\_PERIOD\_ADD;Extension

EVCA\_STRUCTURE;Structure

EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage

EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography

FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV

LAST\_FUNDAMIN\_VALUATION;Last total valuation (fund admin){F}Dernière valorisation totale (admin de fonds)

LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

REMAINING\_COMMITMENT;Remaining commitment{F}Engagement résiduel

REMAINING\_COMMITMENT2;Remaining commitment(Inv curr){F}Engagement résiduel(devise inv)

CASH\_IN;Cash in{F}Cash in

CASH\_IN2;Cash in(Inv curr){F}Cash in(devise inv)

CASH\_OUT;Cash out{F}Cash out

CASH\_OUT2;Cash out(Inv curr){F}Cash out(devise inv)

CAPITAL\_CALL;Capital call{F}Montant appelé

CAPITAL\_CALL2;Capital call(Inv curr){F}Montant appelé(devise inv)

CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)

CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement

CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(Inv curr){F}Montant appelé hors engagement(devise inv)

MANAGEMENT\_FEES;Management fees{F}Frais de gestion

MANAGEMENT\_FEES2;Management fees(Inv curr){F}Frais de gestion(devise inv)

MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement

MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(Inv curr){F}Frais de gestion hors engagement(devise inv)

OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES2;Other fees(Inv curr){F}Autres frais(devise inv)

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement

OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(Inv curr){F}Autres frais hors engagement(devise inv)

DISTRIBUTION;Distributions{F}Distributions

DISTRIBUTION2;Distributions(Inv curr){F}Distributions(devise inv)

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé

RETURN\_OF\_CAPITAL2;Return of capital(Inv curr){F}Capital remboursé(devise inv)

REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

REDRAWABLE\_CAPITAL2;Redrawable call(Inv curr){F}Distribution temporaire(devise inv)

CARRY;Carry{F}Intérêt prioritaire

CARRY2;Carry(Inv curr){F}Intérêt prioritaire(devise inv)

GAIN\_LOSS;Gain/Loss{F}Plus/moins value

GAIN\_LOSS2;Gain/Loss(Inv curr){F}Plus/moins value(devise inv)

INTERESTS;Interests{F}Intérêts

INTERESTS2;Interests(Inv curr){F}Intérêts(devise inv)

DIVIDENDS;Dividends{F}Dividendes

DIVIDENDS2;Dividends(Inv curr){F}Dividendes(devise inv)

OTHER\_INCOME;Other income{F}Autres revenus

OTHER\_INCOME2;Other income(Inv curr){F}Autres revenus(devise inv)

TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions

TRANSACTION\_COSTS2;Transaction costs(Inv curr){F}Coûts des transactions(devise inv)

COST;Cost{F}Prix de revient

COST2;Cost(2){F}Prix de revient(2)

ADJUSTED\_NAV;Adjusted NAV{F}NAV ajustée

ADJUSTED\_NAV2;Adjusted NAV(Inv curr){F}NAV ajustée(devise inv)

EXPL\_STAKE;Explicit stake{F}% détention explicite

EXPL\_STAKE\_NC;Explicit stake NC{F}% détention explicite NC

LAST\_NAV;Last NAV{F}Dernière NAV

LAST\_NAV\_DATE;Last NAV date{F}Dernière date NAV

LAST\_FUNDAMIN\_VALUATION2;Last total valuation (fund admin)(2){F}Dernière valorisation totale (admin de fonds)(2)

COMMITMENT2;Total commitment(Inv curr){F}Engagement total(devise inv)

NAV\_ADJUSTMENT;NAV adjustment{F}Ajustement NAV

CASH\_BALANCE;Cash balance{F}Balance CASH

CASH\_BALANCE2;Cash balance(Inv curr){F}Balance CASH(devise inv)

PERCENT\_CALLED;% called{F}% appelé

PERCENT\_CALLED2;% called(Inv curr){F}% appelé(devise inv)

TVPI;TVPI{F}TVPI

DPI;DPI{F}DPI

RVPI;RVPI{F}RVPI

TVPI2;TVPI(Inv curr){F}TVPI(devise inv)

DPI2;DPI(Inv curr){F}DPI(devise inv)

RVPI2;RVPI(Inv curr){F}RVPI(devise inv)

NET\_IRR;Net IRR{F}TRI NET

NET\_IRR2;Net IRR(Inv curr){F}TRI NET(devise inv)

CASH\_ON\_CASH;Cash on cash{F}Cash on cash

CASH\_ON\_CASH2;Cash on cash(Inv curr){F}Cash on cash(devise inv)

STAKE;% Stake{F}% détention

STAKE\_NC;%Stake{F}% détention

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

LAST\_NAV;Last NAV{F}Dernière NAV

LAST\_NAV\_DATE;Last NAV date{F}Dernière date NAV

COMMITMENT;Total commitment{F}Engagement total

CASH\_ON\_CASH;Cash on cash{F}Cash on cash

CASH\_BALANCE;Cash balance{F}Balance CASH

NAV\_ADJUSTMENT;NAV adjustment{F}Ajustement NAV

PERCENT\_CALLED;% called{F}% appelé

TVPI;TVPI{F}TVPI

DPI;DPI{F}DPI

RVPI;RVPI{F}RVPI

NET\_IRR;Net IRR{F}TRI NET

FUND\_NAME;Fund{F}Fonds

FUND\_ABBREVIATION;Short name{F}Abréviation

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

FUND\_STATUS;Status{F}Statut

FUND\_STATUS\_FORMATTED;Status

FUND\_TYPE;Legal form

FUND\_REGION;Geography

MANAGEMENT\_COMPANY\_ID;IQId

MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion

VINTAGE\_YEAR;Vintage year{F}Millésime

FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing

FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing

CURRENCY;Currency{F}Devise

DOMICILE;Domicile{F}Domiciliation

LEGAL\_FORM;Legal form{F}Forme juridique

FUND\_SIZE;Size{F}Taille

TERM\_YEARS;Term (Years){F}Durée (années)

INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)

TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable

END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement

INVESTMENT\_PERIOD\_ADD;Extension

EVCA\_STRUCTURE;Structure

EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage

EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography

FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV

LAST\_FUNDAMIN\_VALUATION;Last total valuation (fund admin){F}Dernière valorisation totale (admin de fonds)

LAST\_FUNDAMIN\_VALUATION\_DATE;Last valuation date (fund admin){F}Date de dernière valorisation (admin de fonds)

NET\_IRR\_WO\_CARRIED;Net IRR wo Carried{F}TRI NET Hors Carried

## 6.9.17. FV\_T\_FUND\_POSITIONS\_BY\_INVESTORS

### 6.9.17.1. FV\_T\_FUND\_POSITIONS\_BY\_INVESTORS

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

INVESTOR\_CURRENCY;Investor currency{F}Devise investisseur

INVESTOR\_FUND\_ID;ID fund investor{F}ID fonds Investisseur

INVESTOR\_FUND\_NAME;Name fund investor{F}Nom fonds Investisseur

INVESTOR\_FUND\_CURRENCY;Currency fund investor{F}Devise fonds Investisseur

INVESTOR\_FUND\_STATUS;Status fund investor{F}Statut fonds Investisseur

INVESTOR\_COMPANY\_ID;ID company investor{F}ID société Investisseur

INVESTOR\_COMPANY\_NAME;Name company investor{F}Nom société Investisseur

INVESTOR\_COMPANY\_CURRENCY;Currency company investor{F}Devise société Investisseur

INVESTOR\_COMPANY\_STATUS;Status company investor{F}Statut société Investisseur

INVESTOR\_CONTACT\_ID;ID contact investor{F}ID contact Investisseur

INVESTOR\_CONTACT\_LASTNAME;Lastname contact investor{F}Nom contact Investisseur

INVESTOR\_CONTACT\_FIRSTNAME;Firstname contact investor{F}Prénom contact Investisseur

INVESTOR\_CONTACT\_CURRENCY;Currency contact investor{F}Devise contact Investisseur

INVESTOR\_CONTACT\_STATUS;Status contact investor{F}Statut contact Investisseur

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

COMMITMENT2;Total commitment(Inv curr){F}Engagement total(devise inv)

REMAINING\_COMMITMENT2;Remaining commitment(Inv curr){F}Engagement résiduel(devise inv)

CASH\_IN2;Cash in(Inv curr){F}Cash in(devise inv)

CASH\_OUT2;Cash out(Inv curr){F}Cash out(devise inv)

CAPITAL\_CALL2;Capital call(Inv curr){F}Montant appelé(devise inv)

CAPITAL\_CALL\_INSIDE\_COMM2;Capital call inside commitment(2){F}Montant appelé dans l'engagement(2)

CAPITAL\_CALL\_OUTSIDE\_COMM2;Capital call outside commitment(Inv curr){F}Montant appelé hors engagement(devise inv)

MANAGEMENT\_FEES2;Management fees(Inv curr){F}Frais de gestion(devise inv)

MANAGEMENT\_FEES\_OUTSIDE\_COMM2;Management fees outside commitment(Inv curr){F}Frais de gestion hors engagement(devise inv)

OTHER\_FEES2;Other fees(Inv curr){F}Autres frais(devise inv)

OTHER\_FEES\_OUTSIDE\_COMM2;Other fees outside commitment(Inv curr){F}Autres frais hors engagement(devise inv)

DISTRIBUTION2;Distributions(Inv curr){F}Distributions(devise inv)

RETURN\_OF\_CAPITAL2;Return of capital(Inv curr){F}Capital remboursé(devise inv)

REDRAWABLE\_CAPITAL2;Redrawable call(Inv curr){F}Distribution temporaire(devise inv)

CARRY2;Carry(Inv curr){F}Intérêt prioritaire(devise inv)

GAIN\_LOSS2;Gain/Loss(Inv curr){F}Plus/moins value(devise inv)

INTERESTS2;Interests(Inv curr){F}Intérêts(devise inv)

DIVIDENDS2;Dividends(Inv curr){F}Dividendes(devise inv)

OTHER\_INCOME2;Other income(Inv curr){F}Autres revenus(devise inv)

TRANSACTION\_COSTS2;Transaction costs(Inv curr){F}Coûts des transactions(devise inv)

COST2;Cost(2){F}Prix de revient(2)

ADJUSTED\_NAV2;NAV(Inv curr){F}NAV(devise inv)

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

CASH\_BALANCE2;Cash balance(Inv curr){F}Balance CASH(devise inv)

CASH\_ON\_CASH2;Cash on cash{F}Cash on cash

PERCENT\_CALLED2;% called(Inv curr){F}% appelé(devise inv)

TVPI2;TVPI(Inv curr){F}TVPI(devise inv)

DPI2;DPI(Inv curr){F}DPI(devise inv)

RVPI2;RVPI(Inv curr){F}RVPI(devise inv)

NET\_IRR2;Net IRR(Inv curr){F}TRI NET(devise inv)

## 6.9.18. FV\_T\_FUND\_OPERATIONS\_BY\_INVESTORS

### 6.9.18.1. FV\_T\_FUND\_OPERATIONS\_BY\_INVESTORS

OPERATION\_ID;ID fund operation{F}ID opération fonds

INVESTMENT\_ID;ID investment{F}ID Investissement

DRAFT;Draft{F}Brouillon

TRANSACTION\_CANCELLED;Cancelled

FUND\_ID;ID fund{F}ID Fonds

TYPE;Type{F}Type

OPERATION\_TYPE;Formatted type{F}Type formatté

OPERATION\_NAME;Name fund operation{F}Libellé opération fonds

YEAR;Year{F}Année

REFERENCE\_DATE;Date{F}Date

INDEX;Index{F}Index

INVESTOR\_ID;ID investor{F}ID investisseur

INVESTOR\_NAME;Investor name{F}Nom investisseur

REPORT\_DATE;As of Date{F}Date d'arrêté

DATA MART\_DATE;Datamart date{F}Date du datamart

COMMITMENT;Commitment{F}Engagement

CASH\_IN;Cash IN{F}Cash IN

CASH\_OUT;Cash OUT{F}Cash OUT

CAPITAL\_CALL;Capital call{F}Montant appelé

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement

CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement

MANAGEMENT\_FEES;Management fees{F}Frais de gestion

MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement

OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement

DISTRIBUTION;Distributions{F}Distributions

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé

REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

CARRY;Carry{F}Intérêt prioritaire

GAIN\_LOSS;Gain/Loss{F}Plus/moins value

INTERESTS;Interests{F}Intérêts

DIVIDENDS;Dividends{F}Dividendes

OTHER\_INCOME;Other income{F}Autres revenus

TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions

COST;Cost{F}Prix de revient

NAV;NAV{F}NAV

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intêrets hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

CASH\_BALANCE;Cash balance{F}Balance CASH

CASH\_ON\_CASH;Cash on cash{F}Cash on cash

FUND\_NAME;Fund{F}Fonds

FUND\_ABBREVIATION;Short name{F}Abréviation

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

FUND\_STATUS;Status{F}Statut

FUND\_STATUS\_FORMATTED;Status

FUND\_TYPE;Legal form

FUND\_REGION;Geography

MANAGEMENT\_COMPANY\_ID;IQId

MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion

VINTAGE\_YEAR;Vintage year{F}Millésime

FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing

FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing

CURRENCY;Currency{F}Devise

DOMICILE;Domicile{F}Domiciliation

LEGAL\_FORM;Legal form{F}Forme juridique

FUND\_SIZE;Size{F}Taille

TERM\_YEARS;Term (Years){F}Durée (années)

INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)

TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable

END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement

INVESTMENT\_PERIOD\_ADD;Extension

EVCA\_STRUCTURE;Structure

EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage

EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography

FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV

## 6.9.19. FV\_T\_FUND\_OPERATIONS\_BY\_FUNDS

### 6.9.19.1. FV\_T\_FUND\_OPERATIONS\_BY\_FUNDS

OPERATION\_ID;ID fund operation{F}ID opération fonds

DRAFT;Draft{F}Brouillon

TRANSACTION\_CANCELLED;Cancelled

FUND\_ID;ID fund{F}ID Fonds

TYPE;Type{F}Type

OPERATION\_TYPE;Formatted type{F}Type formatté

OPERATION\_NAME;Name fund operation{F}Libellé opération fonds

YEAR;Year{F}Année

REFERENCE\_DATE;Date{F}Date

INDEX;Index{F}Index

REPORT\_DATE;As of Date{F}Date d'arrêté

DATAMART\_DATE;Datamart date{F}Date du datamart

COMMITMENT;Commitment{F}Engagement

CASH\_IN;Cash IN{F}Cash IN

CASH\_OUT;Cash OUT{F}Cash OUT

CAPITAL\_CALL;Capital call{F}Montant appelé

CAPITAL\_CALL\_INSIDE\_COMM;Capital call inside commitment{F}Montant appelé dans l'engagement

CAPITAL\_CALL\_OUTSIDE\_COMM;Capital call outside commitment{F}Montant appelé hors engagement

MANAGEMENT\_FEES;Management fees{F}Frais de gestion

MANAGEMENT\_FEES\_OUTSIDE\_COMM;Management fees outside commitment{F}Frais de gestion hors engagement

OTHER\_FEES;Other fees{F}Autres frais

OTHER\_FEES\_OUTSIDE\_COMM;Other fees outside commitment{F}Autres frais hors engagement

DISTRIBUTION;Distributions{F}Distributions

RETURN\_OF\_CAPITAL;Return of capital{F}Capital remboursé

REDRAWABLE\_CAPITAL;Redrawable call{F}Distribution temporaire

CARRY;Carry{F}Intérêt prioritaire

GAIN LOSS;Gain/Loss{F}Plus/moins value

INTERESTS;Interests{F}Intérêts

DIVIDENDS;Dividends{F}Dividendes

OTHER\_INCOME;Other income{F}Autres revenus

TRANSACTION\_COSTS;Transaction costs{F}Coûts des transactions

COST;Cost{F}Prix de revient

NAV;NAV{F}NAV

OTHER\_INCOME\_OUTSIDE\_OF\_COMMITMENT;Other income outside of commitment{F}Autres revenus hors engagement

GAIN\_LOSS\_OUTSIDE\_OF\_COMMITMENT;Gain or loss outside of commitment{F}Plus/moins values hors engagement

INTEREST\_OUTSIDE\_OF\_COMMITMENT;Interest outside of commitment{F}Intérêts hors engagement

UNSPECIFIED\_DIST\_OUTSIDE\_OF\_COMMITMENT;Unspecified distributions outside of commitment{F}Distributions non spécifiée hors engagement

LATE\_INTEREST\_FEES\_OUTSIDE\_OF\_COMMITMENT;Late interest fees outside of commitment{F}Frais d'intérêt tardif hors engagement

LATE\_INTEREST\_FEES;Late interest fees{F}Frais d'intérêt tardif

UNSPECIFIED\_DISTRIBUTION;Unspecified distribution{F}Distribution non spécifiée

TAX;Tax{F}Taxe

TAX\_REFUNDED;Refunded Tax{F}Taxe remboursé

CASH\_BALANCE;Cash balance{F}Balance CASH

FUND\_NAME;Fund{F}Fonds

FUND\_ABBREVIATION;Short name{F}Abréviation

FUND\_GROUPBY\_ID;Master fund{F}Fonds master

FUND\_STATUS;Status{F}Statut

FUND\_STATUS\_FORMATTED;Status

FUND\_TYPE;Legal form

FUND\_REGION;Geography

MANAGEMENT\_COMPANY\_ID;IQId

MANAGEMENT\_COMPANY;Fund manager{F}Société de gestion

VINTAGE\_YEAR;Vintage year{F}Millésime

FIRST\_CLOSING\_DATE;First closing date{F}Date premier closing

FINAL\_CLOSING\_DATE;Final closing date{F}Date dernier closing

CURRENCY;Currency{F}Devise

DOMICILE;Domicile{F}Domiciliation

LEGAL\_FORM;Legal form{F}Forme juridique

FUND\_SIZE;Size{F}Taille

TERM\_YEARS;Term (Years){F}Durée (années)

INVESTMENT\_PERIOD\_YEARS;Investment period (years){F}Durée d'investissement (années)

TERM\_ADD;Additional year(s){F}Nombre d'année(s) prorogeable

END\_INVESTMENT\_DATE;End investment date{F}Date limite d'investissement

INVESTMENT\_PERIOD\_ADD;Extension

EVCA\_STRUCTURE;Structure

EVCA\_INVESTMENT\_FOCUS\_BY\_STAGE;Investment focus by stage

EVCA\_INVESTMENT\_FOCUS\_BY\_GEOGRAPHY;Investment focus by geography

FUND\_NAVPERIODICITY;Nav periodicity{F}Périodicité de la NAV

## 6.10. Data tables produced by the Consolidated CE (CCE)

The eFront Invest Consolidated calculation engine produces the following tables:

- ▶ [InvestmentPositions](#)

### INVESTMENTPOSITIONS

```
New CECubeColumn() { _  
    New CECubeColumn("POSITION_DATE", "Position Date{F}Date fundposition",  
        GetType(DateTime)), _  
    New CECubeColumn("INVESTEE_ID", "Investee ID{F}ID de l'investi",  
        GetType(String)), _  
    New CECubeColumn("INVESTEEENAME", "Investee name{F}Nom de l'investi",  
        GetType(String)), _  
    New CECubeColumn("INVESTEECURRENCY", "Investee currency{F}Devise de  
        l'investi", GetType(String)), _  
    New CECubeColumn("INVESTEETYPE", "Investee type{F}Type de l'investi",  
        GetType(String)), _  
    New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",  
        GetType(String)), _  
    New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",  
        GetType(String)), _  
    New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",  
        GetType(String)), _  
    New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de  
        l'investisseur", GetType(String)), _  
    New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des  
        montants", GetType(String)), _  
    New CECubeColumn("INVESTMENTEXITDATE", "Exit date{F}Date de sortie",  
        GetType(Date)), _  
    New CECubeColumn("ISEXITED", "Exit?{F}Sorti ?", GetType(Boolean)), _  
    New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",  
        GetType(Decimal)), _  
    New CECubeColumn("REMAININGCOMMITMENT", "Remaining  
        commitment{F}Engagement résiduel", GetType(Decimal)), _
```

```

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), _
New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), _
New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), _
New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",
GetType(Decimal)), _
New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
GetType(Decimal)), _
New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date
NAV", GetType(DateTime)), _
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",
GetType(Decimal)), _
New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)), _
New CECubeColumn("STAKE", "Stake %{F}% détention", GetType(Double)) _
}_
• ► InvestorPositions

```

## INVESTORPOSITIONS

```

New CECubeColumn() {_
New CECubeColumn("POSITION_DATE", "Position Date{F}Date fundposition",
GetType(DateTime)), _
New CECubeColumn("INVESTOR_ID", "Investor ID{F}ID de l'investisseur",
GetType(String)), _
New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",
GetType(String)), _
New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",
GetType(String)), _
New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de
l'investisseur", GetType(String)), _
New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des
montants", GetType(String)), _

```

```
New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",
GetType(Decimal)), _
New CECubeColumn("REMAININGCOMMITMENT", "Remaining
commitment{F}Engagement résiduel", GetType(Decimal)), _
New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), _
New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), _
New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), _
New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",
GetType(Decimal)), _
New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",
GetType(Decimal)), _
New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",
GetType(Decimal)), _
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date
NAV", GetType(DateTime)), _
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",
GetType(Decimal)), _
New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _
New CECubeColumn("IRRREALIZED", "Realized IRR{F}TRI cédé",
GetType(Double)), _
New CECubeColumn("IRRUNREALIZED", "Unrealized IRR{F}TRI non cédé",
GetType(Double)), _
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)) _
} _
```

## 6.10.1. InvestmentPositions

### 6.10.1.1. INVESTMENTPOSITIONS

```
New CECubeColumn() { _
```

```
New CECubeColumn("POSITION_DATE", "Position Date{F}Date fundposition",
GetType(DateTime)), _
```

New CECubeColumn("INVESTEE\_ID", "Investee ID{F}ID de l'investi", GetType(String)), \_

New CECubeColumn("INVESTEEENAME", "Investee name{F}Nom de l'investi", GetType(String)), \_

New CECubeColumn("INVESTEECURRENCY", "Investee currency{F}Devise de l'investi", GetType(String)), \_

New CECubeColumn("INVESTEETYPE", "Investee type{F}Type de l'investi", GetType(String)), \_

New CECubeColumn("INVESTOR\_ID", "Investor ID{F}ID de l'investisseur", GetType(String)), \_

New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur", GetType(String)), \_

New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur", GetType(String)), \_

New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de l'investisseur", GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des montants", GetType(String)), \_

New CECubeColumn("INVESTMENTEXITDATE", "Exit date{F}Date de sortie", GetType(Date)), \_

New CECubeColumn("ISEXITED", "Exit?{F}Sorti ?", GetType(Boolean)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement", GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_

```
New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), _  
New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), _  
New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",  
GetType(Decimal)), _  
New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), _  
New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), _  
New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",  
GetType(Decimal)), _  
New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",  
GetType(Decimal)), _  
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",  
GetType(DateTime)), _  
New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",  
GetType(Decimal)), _  
New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _  
New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)), _  
New CECubeColumn("STAKE", "Stake %{F}% détention", GetType(Double)) _  
}
```

## 6.10.2. InvestorPositions

### 6.10.2.1. INVESTORPOSITIONS

```
New CECubeColumn() { _
```

```
New CECubeColumn("POSITION_DATE", "Position Date{F}Date fundposition",  
GetType(DateTime)), _
```

New CECubeColumn("INVESTOR\_ID", "Investor ID{F}ID de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORNAME", "Investor name{F}Nom de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORTYPE", "Investor type{F}Type de l'investisseur",  
GetType(String)), \_

New CECubeColumn("INVESTORCURRENCY", "Investor currency{F}Devise de  
l'investisseur", GetType(String)), \_

New CECubeColumn("AMOUNTCURRENCY", "Amount currency{F}Devise des  
montants", GetType(String)), \_

New CECubeColumn("COMMITMENT", "Commitment{F}Engagement",  
GetType(Decimal)), \_

New CECubeColumn("REMAININGCOMMITMENT", "Remaining  
commitment{F}Engagement résiduel", GetType(Decimal)), \_

New CECubeColumn("CASH", "Cash balance", GetType(Decimal)), \_

New CECubeColumn("CASHIN", "Cash in", GetType(Decimal)), \_

New CECubeColumn("CASHOUT", "Cash out", GetType(Decimal)), \_

New CECubeColumn("GAINSLOSSES", "Gains/Losses{F}+/- values",  
GetType(Decimal)), \_

New CECubeColumn("INTERESTS", "Interests{F}Intérêts", GetType(Decimal)), \_

New CECubeColumn("DIVIDENDS", "Dividends{F}Dividendes", GetType(Decimal)), \_

New CECubeColumn("OTHERINCOME", "Other income{F}Autres revenus",  
GetType(Decimal)), \_

New CECubeColumn("LASTVALUATION", "Last NAV{F}Dernière NAV",  
GetType(Decimal)), \_

```
New CECubeColumn("LASTVALUATIONDATE", "Last NAV date{F}Dernière date NAV",
GetType(DateTime)), _

New CECubeColumn("ADJUSTEDVALUATION", "Adjusted NAV{F}NAV ajustée",
GetType(Decimal)), _

New CECubeColumn("IRR", "IRR{F}TRI", GetType(Double)), _

New CECubeColumn("IRRREALIZED", "Realized IRR{F}TRI cédé", GetType(Double)),
_

New CECubeColumn("IRRUNREALIZED", "Unrealized IRR{F}TRI non cédé",
GetType(Double)), _

New CECubeColumn("MULTIPLE", "Multiple{F}Multiple", GetType(Double)) _

}
```

## 6.11. Details about calculations

To get details about how values are calculated, refer to the document **FIA - FrontVenture Calculations, calc engines**, which is available for internal use in our documentation catalogue.

## 6.12. Examples - Calculation Engine Tables

### ► DATA - SET - DCE.datatable

#### 6.12.1. Goal

Call the **eFront Invest Funds calculation** engine from within an eFront Script program.

#### 6.12.2. Features being illustrated

- eFront Report libraries
- %DEFINE statement
- %LET statement
- Functions - special
- SET statement

#### 6.12.3. Program

```
//%DEFINE CREATE_TABLES; //uncomment to create  
LIBNAME VIEWS "\Views";  
%DEFINE REPORT_DATE;  
  
%PARAM DATE LABEL="Reference date{F}Date d'arrêté" TYPE=DATE  
DEFAULT=TODAY NOTNULL;  
  
%PARAM IQID_FUND LABEL="Fund{F}Fonds" TYPE=STRING INFORMAT="??  
XPICKID(SELECT DISTINCT A.IQID, A.FUND FROM VCFUND A INNER JOIN  
VCINVESTMENT B ON B.FUND=A.IQID WHERE ??FILTER ORDER BY A.FUND);  
  
%PARAM IQID_COMPANY LABEL="Company{F}Société" TYPE=STRING  
INFORMAT="??XPICKID(SELECT DISTINCT A.IQID, A.NAME FROM SFAACCOUNT  
A INNER JOIN VCINVESTMENT B ON B.FIRM=A.IQID WHERE ??FILTER ORDER BY  
A.NAME);  
  
%LET POSITION_DATE = "Positions as of " & CDATE(%DATE);
```

```
%IF EXISTMACRO("CREATE_TABLES") %THEN  
  
LIBNAME USER ".":;  
  
%LET IQID_FUND = NOTHING;  
  
%LET IQID_FUND = "NONE";  
  
%ELSE  
  
LIBNAME USER WORK; // Tables created in memory  
  
%END;  
  
//-----  
  
// Direct portfolio  
  
//-----  
  
%DEFINE PORTFOLIO_REPORT;  
  
// %DEFINE USE_INVESTEE_CURR; // uncomment to use companies currencies  
  
// %DEFINE USE_DRAFT; // uncomment to use draft transactions  
  
%DEFINE CALCULATE SHARES; // to calculate ND/FD ownership  
  
%DEFINE CALCULATE_ACCRUALS; // To call , for PIK / Cash accruals and adjusted  
valuation if necessary  
  
%DEFINE CALCULATE_IRR; // to calculate IRR  
  
//%LET IQID_INVESTOR = %IQID_FUND;  
  
// Fund direct positions in invested portfolio companies  
  
DATA T_DIRECT_INVESTMENT_POSITIONS(where INVESTMENT_ID<>""); // to  
exclude fees and cash instruments (= internal investment)
```

```
SET DCE.INVESTMENTPOSITIONS(WHERE INVESTOR_ID IN (%IQID_FUND) OR  
INVESTOR_ID IN (%IQID_COMPANY));
```

```
RUN;
```

```
%LET COMPANY_LIST = COLLECTION("T_DIRECT_INVESTMENT_POSITIONS",  
"COMPANY_ID");
```

```
%IF EXISTMACRO("CREATE_TABLES") %THEN
```

```
%LET COMPANY_LIST = NOTHING;
```

```
%LET COMPANY_LIST = "NONE";
```

```
%END;
```

```
DATA T_DIRECT_INVESTMENT_POSITIONS;
```

```
MERGE T_DIRECT_INVESTMENT_POSITIONS(IN=FLAG) VIEWS.V_COMPANY;
```

```
BY COMPANY_ID;
```

```
COLUMN INDUSTRY_LABEL;
```

```
INDUSTRY_LABEL = LOOKUP("VCINDUSTRY", INDUSTRY);
```

```
_OUTPUT_=FLAG;
```

```
RUN;
```

▶ [DATA - SET - FCE.datatable](#)

#### 6.12.4. Goal

Call the **eFront Invest Funds calculation** engine from within an eFront Script program.

#### 6.12.5. Features being illustrated

- [eFront Report libraries](#)

- %DEFINE statement
- %LET statement
- Functions - special
- SET statement

### 6.12.6. Program

```
LIBNAME USER ".";  
  
%DEFINE REPORT_DATE;  
  
%LET DATE=TODAY();  
  
DATA WORK.list;  
  
SQL "select IQID from vcfund A where ??FILTER";  
  
COLUMN IQID;  
  
RUN;  
  
DATA WORK.list;  
  
SET WORK.list(from=1 to=100);  
  
RUN;  
  
%LET iqid_fof_fund = COLLECTION("work.list", "iqid");  
  
//%let iqid_fof_fund = "E8AB57217FD948CE9B44E9BBB4D7AF9E"; // A11  
  
%DEFINE FOF_INVESTOR_REPORT;  
  
DATA WORK.tmp1;  
  
SET FCE.FUNDOPERATIONS;  
  
RUN;
```

► DATA - SET - CCE.datatable

## 6.12.7. Goal

Call the **eFront Invest Funds calculation** engine from within an eFront Script program.

## 6.12.8. Features being illustrated

- [eFront Report libraries](#)
- [%DEFINE statement](#)
- [%LET statement](#)
- [Functions - special](#)
- [SET statement](#)

## 6.12.9. Program

```
//%DEFINE CREATE_TABLES; //uncomment to create  
LIBNAME VIEWS "\Views";  
%DEFINE REPORT_DATE;  
  
%PARAM DATE LABEL="Reference date{F}Date d'arrêté" TYPE=DATE  
DEFAULT=TODAY NOTNULL;  
  
%PARAM IQID_FUND LABEL="Fund{F}Fonds" TYPE=STRING INFORMAT=?  
XPICKID(SELECT DISTINCT A.IQID, A.FUND FROM VCFUND A LEFT JOIN  
VCINVESTMENT B ON B.FUND=A.IQID LEFT JOIN VCSUBSCRVEHICUL C ON  
C.FUND=A.IQID WHERE ??FILTER AND (B.IQID IS NOT NULL OR C.IQID IS NOT  
NULL) ORDER BY FUND");  
  
%PARAM IQID_COMPANY LABEL="Company{F}Société" TYPE=STRING  
INFORMAT=?XPICKID(SELECT DISTINCT A.IQID, A.NAME FROM SFAACCOUNT  
A LEFT JOIN VCINVESTMENT B ON B.FIRM=A.IQID LEFT JOIN  
VCSUBSCRVEHICUL C ON C.ACCOUNT=A.IQID WHERE ??FILTER AND (B.IQID IS  
NOT NULL OR C.IQID IS NOT NULL) ORDER BY NAME");  
  
%LET POSITION_DATE = "Positions as of " & CDATE(%DATE);  
  
%IF EXISTMACRO("CREATE_TABLES") %THEN
```

```
LIBNAME USER ".";
%LET IQID_FUND = NOTHING;
%LET IQID_FUND = "NONE";
%ELSE
LIBNAME USER WORK; // Tables created in memory
%END;
//-----
// Direct portfolio
//-----
//%LET IQID_INVESTOR = %IQID_FUND;
// Fund direct positions in invested portfolio companies
DATA T_CONSOLIDATED_INVESTMENT_POSITIONS;
SET CCE.INVESTMENTPOSITIONS(WHERE INVESTOR_ID IN (%IQID_FUND) OR
INVESTOR_ID IN (%IQID_COMPANY));
RUN;
DATA T_CONSOLIDATED_INVESTOR_POSITIONS;
SET CCE.INVESTORPOSITIONS(WHERE INVESTOR_ID IN (%IQID_FUND) OR
INVESTOR_ID IN (%IQID_COMPANY));
RUN;
//proc print DATA=T_CONSOLIDATED_INVESTMENT_POSITIONS;run;
//proc print DATA=T_CONSOLIDATED_INVESTOR_POSITIONS;run;
```

## 6.12.10. DATA - SET - CCE.datatable

### 6.12.10.1. Goal

Call the **eFront Invest Funds calculation** engine from within an eFront Script program.

### 6.12.10.2. Features being illustrated

- eFront Report libraries
- %DEFINE statement
- %LET statement
- Functions - special
- SET statement

### 6.12.10.3. Program

```
//%DEFINE CREATE_TABLES; //uncomment to create  
  
LIBNAME VIEWS "\Views";  
  
%DEFINE REPORT_DATE;  
  
%PARAM DATE LABEL="Reference date{F}Date d'arrêté" TYPE=DATE  
DEFAULT=TODAY NOTNULL;  
  
%PARAM IQID_FUND LABEL="Fund{F}Fonds" TYPE=STRING INFORMAT="??  
XPICKID(SELECT DISTINCT A.IQID, A.FUND FROM VCFUND A LEFT JOIN  
VCINVESTMENT B ON B.FUND=A.IQID LEFT JOIN VCSUBSCRVEHICUL C ON  
C.FUND=A.IQID WHERE ??FILTER AND (B.IQID IS NOT NULL OR C.IQID IS NOT  
NULL) ORDER BY FUND");  
  
%PARAM IQID_COMPANY LABEL="Company{F}Société" TYPE=STRING  
INFORMAT="??XPICKID(SELECT DISTINCT A.IQID, A.NAME FROM SFAACCOUNT  
A LEFT JOIN VCINVESTMENT B ON B.FIRM=A.IQID LEFT JOIN  
VCSUBSCRVEHICUL C ON C.ACCOUNT=A.IQID WHERE ??FILTER AND (B.IQID IS  
NOT NULL OR C.IQID IS NOT NULL) ORDER BY NAME");  
  
%LET POSITION_DATE = "Positions as of " & CDATE(%DATE);
```

```
%IF EXISTMACRO("CREATE_TABLES") %THEN  
LIBNAME USER ".";  
  
%LET IQID_FUND = NOTHING;  
  
%LET IQID_FUND = "NONE";  
  
%ELSE  
  
LIBNAME USER WORK; // Tables created in memory  
  
%END;  
  
//-----  
  
// Direct portfolio  
  
//-----  
  
//%LET IQID_INVESTOR = %IQID_FUND;  
  
// Fund direct positions in invested portfolio companies  
  
DATA T_CONsolidated_inVESTMENT_POSITIONS;  
  
SET CCE.inVESTMENTPOSITIONS(WHERE INVESTOR_ID IN (%IQID_FUND) OR  
INVESTOR_ID IN (%IQID_COMPANY));  
  
RUN;  
  
DATA T_CONsolidated_inVESTOR_POSITIONS;  
  
SET CCE.inVESTORPOSITIONS(WHERE INVESTOR_ID IN (%IQID_FUND) OR  
INVESTOR_ID IN (%IQID_COMPANY));  
  
RUN;  
  
//proc print DATA=T_CONsolidated_inVESTMENT_POSITIONS;run;  
  
//proc print DATA=T_CONsolidated_inVESTOR_POSITIONS;run;
```

## 6.12.11. DATA - SET - DCE.datatable

### 6.12.11.1. Goal

Call the **eFront Invest Funds calculation** engine from within an eFront Script program.

### 6.12.11.2. Features being illustrated

- [eFront Report libraries](#)
- [%DEFINE statement](#)
- [%LET statement](#)
- [Functions - special](#)
- [SET statement](#)

### 6.12.11.3. Program

```
//%DEFINE CREATE_TABLES; //uncomment to create  
  
LIBNAME VIEWS "\Views";  
  
%DEFINE REPORT_DATE;  
  
%PARAM DATE LABEL="Reference date{F}Date d'arrêté" TYPE=DATE  
DEFAULT=TODAY NOTNULL;  
  
%PARAM IQID_FUND LABEL="Fund{F}Fonds" TYPE=STRING INFORMAT="??  
XPICKID(SELECT DISTINCT A.IQID, A.FUND FROM VCFUND A INNER JOIN  
VCINVESTMENT B ON B.FUND=A.IQID WHERE ??FILTER ORDER BY A.FUND)";  
  
%PARAM IQID_COMPANY LABEL="Company{F}Société" TYPE=STRING  
INFORMAT="??XPICKID(SELECT DISTINCT A.IQID, A.NAME FROM SFAACCOUNT  
A INNER JOIN VCINVESTMENT B ON B.FIRM=A.IQID WHERE ??FILTER ORDER BY  
A.NAME)";  
  
%LET POSITION_DATE = "Positions as of " & CDATE(%DATE);  
  
%IF EXISTMACRO("CREATE_TABLES") %THEN  
  
LIBNAME USER ":";
```

```
%LET IQID_FUND = NOTHING;  
  
%LET IQID_FUND = "NONE";  
  
%ELSE  
  
LIBNAME USER WORK; // Tables created in memory  
  
%END;  
  
//-----  
  
// Direct portfolio  
  
//-----  
  
%DEFINE PORTFOLIO_REPORT;  
  
// %DEFINE USE_INVESTEE_CURR; // uncomment to use companies currencies  
  
// %DEFINE USE_DRAFT; // uncomment to use draft transactions  
  
%DEFINE CALCULATE SHARES; // to calculate ND/FD ownership  
  
%DEFINE CALCULATE_ACCRUALS; // To call , for PIK / Cash accruals and adjusted  
valuation if necessary  
  
%DEFINE CALCULATE_IRR; // to calculate IRR  
  
//%LET IQID_INVESTOR = %IQID_FUND;  
  
// Fund direct positions in invested portfolio companies  
  
DATA T_DIRECT_INVESTMENT_POSITIONS(where INVESTMENT_ID<>""); // to  
exclude fees and cash instruments (= internal investment)  
  
SET DCE.INVESTMENTPOSITIONS(WHERE INVESTOR_ID IN (%IQID_FUND) OR  
INVESTOR_ID IN (%IQID_COMPANY));  
  
RUN;
```

```
%LET COMPANY_LIST = COLLECTION("T_DIRECT_INVESTMENT_POSITIONS",
"COMPANY_ID");

%IF EXISTMACRO("CREATE_TABLES") %THEN

%LET COMPANY_LIST = NOTHING;

%LET COMPANY_LIST = "NONE";

%END;

DATA T_DIRECT_INVESTMENT_POSITIONS;

MERGE T_DIRECT_INVESTMENT_POSITIONS(IN=FLAG) VIEWS.V_COMPANY;

BY COMPANY_ID;

COLUMN INDUSTRY_LABEL;

INDUSTRY_LABEL = LOOKUP("VCINDUSTRY", INDUSTRY);

_OUTPUT_=FLAG;

RUN;
```

## 6.12.12. DATA - SET - FCE.datatable

### 6.12.12.1. Goal

Call the **eFront Invest Funds calculation** engine from within an eFront Script program.

### 6.12.12.2. Features being illustrated

- [eFront Report libraries](#)
- [%DEFINE statement](#)
- [%LET statement](#)
- [Functions - special](#)
- [SET statement](#)

### 6.12.12.3. Program

```
LIBNAME USER ".";  
  
%DEFINE REPORT_DATE;  
  
%LET DATE=TODAY();  
  
DATA WORK.list;  
  
SQL "select IQID from vcfund A where ??FILTER";  
  
COLUMN IQID;  
  
RUN;  
  
DATA WORK.list;  
  
SET WORK.list(from=1 to=100);  
  
RUN;  
  
%LET iqid_fof_fund = COLLECTION("work.list", "iqid");  
  
//%let iqid_fof_fund = "E8AB57217FD948CE9B44E9BBB4D7AF9E"; // A11  
  
%DEFINE FOF_INVESTOR_REPORT;  
  
DATA WORK.tmp1;  
  
SET FCE.FUNDOPERATIONS;  
  
RUN;
```

## 7. Managing systematic access to external databases

It is possible to query systematically an external database when executing SQL queries contained in eFront Report programs. To do so, you need to indicate the access path to the external database in the **efront.config** file, and create named connections:

[Define access to an external database](#)

## 7.1. Define access to an external database

You can create named connections in the `efront.config` file, that can be referenced from within eFront Report programming steps, and orientate SQL queries systematically towards the external database pointed to by the names connection.

💡 Use this option very cautiously! Giving direct access to external databases skips security facilities, and may result in erroneous database table destruction or manipulation.

1. Access the **efront.config** file.

The file is located on the server, in the `\efront\website` folder.

2. Access the `<db> ... </db>` tag.

3. Use the `<NamedConnections> ... </NamedConnections>` tags to create a named connection.

Here is an example:

```
<db>
  <ConnectionString type="string">Data Source=EF FRONT-1-00193\SQLEXPRESS;User ID=sa;Password=sa</ConnectionString>
  <ADO.NET type="bool">false</ADO.NET>
  <NamedConnections>
    <Connection1>
      <Name type="string">exportdatamart</Name>
      <ConnectionString type="string">Data Source=EF FRONT-1-00193\SQLEXPRESS;User ID=sa;Password=sa</ConnectionString>
    </Connection1>
  </NamedConnections>
</db>
```

4. Save the configuration file.

## 7.2. Accessing eFront Data Warehouse data using eFront Script

If you have the eFront Data Warehouse installed, you can use eFront Script and eFront Cube programs to retrieve, manipulate and create reports based on eFront Data Warehouse data. The data retrieval process is identical to that for other external databases, where data from the database is extracted using SQL queries and imported into eFront Report tables. The data can then be processed, output into different file types, or exported back to the eFront Data Warehouse.

### 7.2.1. Configuring access to the eFront Data Warehouse

Unlike other external databases, the eFront Data Warehouse does not require any additional changes to be made to the connection settings in order to access it with a eFront Script or eFront Cube program. If the eFront Data Warehouse is installed and deployed, its connection string is automatically defined and no further configuration is necessary.

### 7.2.2. Importing and exporting eFront Data Warehouse data

Since the eFront Data Warehouse is treated as an external database, the CONNECTION option must be defined when accessing the eFront Data Warehouse. In the applicable steps and statements, the option needs to be defined as follows:

CONNECTION="DWH"

To import eFront Data Warehouse data into a eFront Report table, the following steps and statements can be used:

Name	Description	Example
PROC SQLIMPORT	Imports eFront Data Warehouse data into an eFront Report table. Can import an entire eFront Data Warehouse table or combine data from several eFront Data Warehouse tables by executing an SQL statement.	PROC SQLIMPORT DATA=WORK.DWH CONNECTION="DWH" TABLE="DWH_Fund" RUN; PROC PRINT DATA=WORK.DWH; RUN;

DATA SQL	Creates an eFront Report table and populates it with data obtained by executing an SQL statement.	<pre>DATA WORK.DWH; SQL "SELECT Name, Abbreviation FROM DWH_Fund A WHERE ??FILTERDWH" CONNECTION="DWH"; COLUMN Name; COLUMN Abbreviation; RUN; PROC PRINT DATA=WORK.DWH; RUN;</pre>
----------	---	---

To export data from an eFront Report table to the eFront Data Warehouse, the following step can be used:

Name	Description	Example
PROC SQLTABLE	Exports data from an eFront Report table into an eFront Data Warehouse table.	<pre>PROC SQLTABLE DATA=WORK.DWH CONNECTION="DWH" TABLE="DWH_NewTable" CREATE RUN;</pre>

You can also execute SQL queries stored in the eFront Data Warehouse with the following step:

Name	Description	Example
PROC SQLEXEC	Launches the execution of an SQL query stored on a server.	<pre>PROC SQLEXEC SQL="DWH_TestProcedure" CONNECTION="DWH"; RUN;</pre>

### 7.2.3. Filtering eFront Data Warehouse data

It is possible to use region filtering for eFront Data Warehouse data in eFront Script, to ensure that users can retrieve only the data from the region that they have access to.

Depending on the method used to retrieve eFront Data Warehouse data, region filtering can be applied either with the **FILTER** option or with the **??FILTERDWH** clause, which are used as follows:

Name	Description	Example
FILTER option	<p>Used in: PROC SQLIMPORT</p> <p>Applied as an option within the step.</p> <p>Available for eFront Script and eFront Cube.</p>	<pre>PROC SQLIMPORT DATA=WORK.DWH CONNECTION="DWH" TABLE="DWH_Fund" FILTER; RUN;  PROC PRINT DATA=WORK.DWH;RUN;</pre>
?? FILTERDWH clause	<p>Used in: DATA SQL, PROC SQLIMPORT</p> <p>Applied within an SQL query.</p> <p>Note that it is necessary to specify an alias when using the clause, for example "DWH_FUND A". The default alias is A, but a specific alias can also be used.</p> <p>Available for eFront Script and eFront Cube.</p> <p> Does not work for the following statement:  <code>%LET "Variable_name"=SQL("SQL_query")</code></p>	<pre>%LET SQL_QUERY = "SELECT Name, RegionID FROM DWH_Fund A where ??FILTERDWH";  PROC SQLIMPORT DATA=WORK.T_OUTPUT SQL=%SQL_QUERY CONNECTION="DWH"; RUN;  PROC PRINT DATA = WORK.T_OUTPUT; RUN;</pre>

### 7.2.3.1. Notes

If the FILTER option or the ??FILTERDWH clause is not used, region filtering has to be done in the eFront Data Warehouse stored procedures.

Region filtering in the eFront Invest database is done with the ??FILTER clause. Like the ??FILTERDWH clause, ??FILTER requires you to specify an alias for the table you want to filter.

## 8. Error debugging

**eFront Script** performs error processing during both program **compilation** and **execution**. **eFront Script** recognizes 2 types of errors:

- **Syntax errors**: the language element does not conform to **eFront Script** language rules (ex.: misspelled eFront Script keyword, unmatched quotation marks, missing semicolon, invalid statement option, invalid table option, ...); error detected at compilation. **eFront Script** prefixes notification messages about syntax errors with a numeric code.
- **Run-time errors**: compilation has been successful, but execution fails (ex.: illegal arguments to functions, illegal mathematical operations, variables in the wrong order for BY-group processing, no data, ...), error detected at execution.

To debug your programs, use different strategies:

- Write **trace marks** to the **LOG** window, using the [TRACE statement](#).
- Activate/Deactivate TRACE statements using the macro variable [%DEBUG](#).

## 8.1. Error messages

### 8.1.1. '=' expected

May indicate that the equal sign is missing when a value should be assigned to a column. But it may indicate also, that a syntax error of a different kind occurred.

### 8.1.2. 0001 DATA\_NONE

No data has been assigned to table.

### 8.1.3. Cannot find Office template file

The template file name is invalid, or the file cannot be found at the location being specified. Most likely to occur when using the [TEMPLATE statement](#).

### 8.1.4. Cannot mix column and statistic at same level

This error is most likely to occur when using the [TABLE statement](#). It might identify an erroneous use of column/line identifiers.

### 8.1.5. Cannot register table

The table that should be read, cannot be found. Most likely to occur when using the [SET statement](#) or the [MERGE statement](#).

### 8.1.6. Column defined more than once in a format clause

The format of a column is being defined twice. More likely to occur when using the [FORMAT statement](#).

### 8.1.7. column is not defined in table

The column being referenced is not defined in the data table. Most likely to occur when merging different tables, and using the [BY statement](#).

## 8.1.8. ERREUR: Option de variable invalide :

The option used to closer define the statement, is invalid. Likely to occur when using the [VAR statement](#).

## 8.1.9. ERROR(0105): {EOF} expected

## 8.1.10. ERROR(0106): '(' expected

An opening parenthesis might have been omitted. Most likely to occur when using [table options](#).

## 8.1.11. ERROR(0107): ')' expected

This error may indicate that a closing parenthesis is missing. It may indicate as well, that something is wrong with the parameters enclosed between the parenthesis.

## 8.1.12. ERROR(0111): ';' expected

"%;" has been left out.

## 8.1.13. ERROR(0113): {Column name} expected

This message is likely to occur when a statement requires a column name as argument, such as the [BY statement](#), the [CLASS statement](#), the [PIE statement](#), [VAR statement](#) ...

## 8.1.14. ERROR(0113): {Format name} expected

The format name might be left out in the [PICTURE statement](#) or the [VALUE statement](#).

## 8.1.15. ERROR(0113): {prefix} expected

Most likely to occur when using the [PROC TRANPOSE step](#). Either the PREFIX value is missing, or it is invalid. If enclosed in quotation strings, the PREFIX value is invalid!

## 8.1.16. ERROR(0113): 'RUN' expected

A RUN statement is expected. Occurs when a statement is not correct, or invalid in a programming step.

## 8.1.17. ERROR(0113): Stylesheet name expected

No stylesheet name has been assigned to the [STYLE option](#) when used as table option.

## 8.1.18. ERROR(0113): Table name expected

The table name has been left out in a statement that requires a table as argument, such as the [SET statement](#) or the [MERGE statement](#). It could be, that the table name has been mentioned but assigned to the statement using an equal sign, where no such sign is expected.

## 8.1.19. ERROR(0113): 'THEN' expected

theN keyword may be missing or being misspelled. It could be too, that the error is due to an erroneous use of the conditional operator (AND, OR,...). Most likely to occur when using the [IF...THEN...ELSE control](#).

## 8.1.20. ERROR(0113): Variable name expected

A variable name is expected. Can occur when using the [IN flag](#) in an erroneous way.

## 8.1.21. ERROR(0121): {filename} expected

A file name is missing, or invalid. Most likely to occur when using the [INFILE statement](#).

## 8.1.22. ERROR(0121): {string} expected

The value assigned to BOX is wrong. Note that there is no equal sign between BOX and the character string. Most likely to occur when using the [BOX statement](#) within a PROC TABULATE step.

## 8.1.23. ERROR(0121): {template-name} expected

The template-name is invalid or missing. Most likely to appear when using the [TEMPLATE statement](#).

## 8.1.24. ERROR(0123): Missing function parameter(s)

The function being used has not enough parameters. Most likely to occur when using [functions](#).

## 8.1.25. ERROR(0123): Too many function parameters

The function being used has too many parameters. Most likely to occur when using [functions](#).

## 8.1.26. ERROR(0127): TOP value expected

The value assigned to the TOP option is incorrect. Most likely to occur when using the [PROC SORT step](#).

## 8.1.27. ERROR(0130): '.' expected

The '.' is missing. Most likely to occur when using the [FORMAT option](#), or the [FORMAT statement](#).

## 8.1.28. ERROR: Chart type not defined

The type of graphic has not been defined. Most likely to occur when using the [PROC GCHART step](#) without the [PIE statement](#).

## 8.1.29. ERROR: Column name defined twice

The column name has been defined twice. Most likely to occur when using the [INLINE statement](#), and feeding values into the data table.

## 8.1.30. ERROR: Format defined twice

The format or picture has been defined more than once. Likely to occur when using the [VALUE statement](#) or the [PICTURE statement](#).

## 8.1.31. ERROR: Invalid definition

The picture definition is wrong. Most likely to occur when using the [PICTURE statement](#) or the [VALUE statement](#).

### 8.1.32. ERROR: Invalid proc option

One of the step options is invalid. Most likely to occur when using the [PROC PRINT](#) step.

### 8.1.33. ERROR: Invalid style element

The item to be styled may be invalid. Most likely to appear when using the [STYLE](#) option or the [STYLESHEET](#) step.

### 8.1.34. ERROR: Invalid style parameter

The style option is not valid. The style attribute may be incorrect or the item to be styled may be invalid. Most likely to appear when using the [STYLE](#) option or the [STYLESHEET](#) step.

### 8.1.35. ERROR: Missing or duplicate column name

The name of the column is not specified, or has already been used. Most likely to occur when defining table columns, for example, when using the [COLUMN](#) statement.

### 8.1.36. ERROR: No BY column defined

The BY statement is missing. This message is likely to occur when using [PROC SORT](#) without a BY statement.

### 8.1.37. invalid PROC qualifier

Most likely to occur when using the [PROC EXCEL](#) step.

### 8.1.38. ERROR: Syntax error in data options definition!

Identifies a syntax error while using [table options](#).

### 8.1.39. ERROR: WHEN, OTHERWISE or END expected

the keywords WHEN, OTHERWISE or END are expected. Most likely to occur when using the [SELECT...WHEN...THEN](#) group.

## 8.1.40. Format not found

The format cannot be identified. Most likely to occur when using the [FORMAT statement](#) or [FORMAT option](#).

## 8.1.41. Include object not found

The object to include couldn't be located. Most likely to occur when using the [%INCLUDE statement](#).

## 8.1.42. 'INTEGER' expected after LENGTH

The value assigned to LENGTH must be an integer. Occurs when using the [LENGTH option](#) while defining a table column.

## 8.1.43. Invalid ALIGN value

The value assigned to ALIGN is invalid. Occurs when using the [ALIGN option](#) while defining a table column.

## 8.1.44. Invalid name

Name invalid. Is likely to occur when using an erroneous option in the [COLUMN statement](#).

## 8.1.45. Invalid Office template

The file being defined is not a valid OFFICE template. Most likely to occur when using the [TEMPLATE statement](#).

## 8.1.46. Invalid or missing COLUMN macro parameter.

A parameter of the [COLUMN statement](#) is invalid or missing. An invalid format might have been used for example.

## 8.1.47. Invalid or unknown library

The library cannot be identified. Most likely to occur when using the [LIBNAME statement](#).

## 8.1.48. Invalid path or path not found

The library (folder) referred to does not exist. This is likely to occur when using the [LIBNAME statement](#).

## 8.1.49. No calc variable defined

Conditions prior to a successful completion of the TABLE statement are not fulfilled. It is possible that the variable identified in the [VAR statement](#) is wrong, or that the calculation variable has been left out in the [TABLE statement](#).

## 8.1.50. Invalid TYPE value

Signals an invalid Data Type. Most likely to occur when using the [TYPE option](#).

## 8.1.51. Invalid value in inline data

The column type and the value are incompatible. Most likely to occur when using the [INLINE statement](#).

## 8.1.52. Office template file is not defined

No OFFICE template (.xlt or .dot) file has been defined. Most likely to occur when using the [TEMPLATE statement](#).

## 8.1.53. Stylesheet not found

Impossible to identify the stylesheet. Most likely to appear when using the [STYLE statement](#).

## 8.1.54. 'TO' expected

The TO keyword is missing. Most likely to occur when using the [FOR...TO...NEXT control](#).

## 8.1.55. Unknown BY column

The column used together with the BY statement is erroneous. Most likely to occur when using the [BY statement](#).

## 8.1.56. Unknown CLASS column

One of the CLASS statement parameters is wrong. Most likely to occur when using the [CLASS statement](#).

## 8.1.57. Unknown IRR column

The column referred to by the [IRR statement](#) is invalid.

## 8.1.58. Unknown MAX column

The column referred to by the [MAX statement](#) is invalid.

## 8.1.59. Unknown MEAN column

The column referred to by the [MEAN statement](#) is invalid.

## 8.1.60. Unknown MIN column

The column referred to by the [MIN statement](#) is invalid.

## 8.1.61. Unknown MULTIPLE column

The column referred to by the [MULTIPLE statement](#) is invalid.

## 8.1.62. Unknown N column

One of the N statement parameters is wrong. Most likely to occur when using the [N statement](#).

## 8.1.63. Unknown name

The column name being used is not known. Can occur when using [loop counters](#) that have not been declared as columns beforehand.

## 8.1.64. Unknown SUM column

The column used together with the SUM statement is erroneous. Most likely to occur when using the [SUM statement](#).

## 8.1.65. Unknown TABLE column

One of the TABLE statement parameters is not recognized. Most likely to occur when using the [TABLE statement](#).

## 9. Dictionary of <FrontScript> tokens

## 9.1. %

Find below the list of **eFront Script** tokens. If you want to know the program step or program statement a particular token is linked at, please use the Research tool of the online help, type the token, and search for it.

## 9.2. %CONTINUE

### 9.3. %DO

## 9.4. %ELSE

## 9.5. %ELSEIF

## 9.6. %END

## 9.7. %EXIT %DO

## 9.8. %EXIT %FOR

## 9.9. %FOR

## 9.10. %IF

## 9.11. %INCLUDE

## 9.12. %LET

## 9.13. %LOOP

## 9.14. %NEXT

## 9.15. %PARAM

## 9.16. %THEN

## 9.17. %TO

## 9.18. %WHILE

## 9.19. ??CHECK

## 9.20. ??CHOICE

## 9.21. ??FILTER

## 9.22. ??ID

## 9.23. ??LOOKUP

## 9.24. ??LOOKUPCODE

## 9.25. ??PICKID

## 9.26. ??XID

## 9.27. ??XLOOKUP

## 9.28. ??XLOOKUPCODE

## 9.29. ??XPICKID

## 9.30. LEVEL

## 9.31. OUTPUT

## 9.32. URL

## 9.33. ABORT

## 9.34. ALIGN

## 9.35. ALL

## 9.36. AMOUNT

## 9.37. AND

## 9.38. ATTACHMENTS

## 9.39. BANNER

## 9.40. BANNERFILE

## 9.41. BCC

## 9.42. BETWEEN

## 9.43. BLOB

## 9.44. BODY

## 9.45. BOOLEAN

## 9.46. BOX

## 9.47. BY

## 9.48. BYTE

## 9.49. CALCULATE\_SHARES

## 9.50. CC

## 9.51. CCE

## 9.52. CE\_CURRENCY

## 9.53. CHAR

## 9.54. CLASS

## 9.55. CLOSING

## 9.56. COLHEADER

## 9.57. COLLAPSE

## 9.58. COLUMN

## 9.59. COLUMNS

## 9.60. COMMA

## 9.61. CONTINUE

## 9.62. DATA

## 9.63. DATE

## 9.64. DCE

## 9.65. DEFAULT

## 9.66. DELIMITER

## 9.67. DESCENDING

## 9.68. DETAILS

## 9.69. DIRECTCASHFLOWS

## 9.70. DO

## 9.71. DONUT

## 9.72. DONUT3D

## 9.73. DOUBLE

## 9.74. DRIVER

## 9.75. DROP

## 9.76. EF\_ATTRIB

## 9.77. EF\_CASE

## 9.78. EF\_G

## 9.79. EF\_LOOP

## 9.80. EF\_MACRO

## 9.81. EF\_OTHERWISE

## 9.82. EF\_SELECT

## 9.83. EF\_SELECT

## 9.84. EF\_VALUE

## 9.85. EFRONTACCRLS

## 9.86. EFRONTBLOB

## 9.87. EFRONTFOLDER

## 9.88. EFRONTIMPORT

## 9.89. EFRONTIMPORT\_AJX

## 9.90. EFRONTMAIL

## 9.91. EFRONTMENUS

## 9.92. EFRONTPACKAGES

## 9.93. EFRONTPROFILES

## 9.94. EFRONTREGIONS

## 9.95. EFRONTTABLES

## 9.96. ELSE

## 9.97. ELSEIF

## 9.98. EMPTY

## 9.99. END

## 9.100. EVENT

## 9.101. EXCEL

## 9.102. EXIT DO

## 9.103. EXIT FOR

## 9.104. EXPAND

## 9.105. EXPORT

## 9.106. EXPORTEXCEL

## 9.107. EXTEND

## 9.108. FALIBRARY

## 9.109. FALSE

## 9.110. FAQUERY

## 9.111. FCE

## 9.112. FILE

## 9.113. FILES

## 9.114. FILTER

## 9.115. FIRST

## 9.116. FIRSTOBS

## 9.117. FLOAT

## 9.118. FLOW

## 9.119. FOR

## 9.120. FORMAT

## 9.121. FREEZEHEADER

## 9.122. FREEZELEFT

## 9.123. FROM

## 9.124. FUNDCASHFLOWS

## 9.125. GCHART

## 9.126. GPLOT

## 9.127. GRANDTOTAL

## 9.128. GROUPFORMAT

## 9.129. HBAR

## 9.130. HBAR3D

## 9.131. HEADER

## 9.132. HORIZONTALLY

## 9.133. HTML

## 9.134. HTMLBODY

## 9.135. ID

## 9.136. IDLABEL

## 9.137. IF

## 9.138. IN

## 9.139. INDEXES

## 9.140. INFILE

## 9.141. INFO

## 9.142. INFORMAT

## 9.143. INLINE

## 9.144. INSTRUMENTS

## 9.145. INTEGER

## 9.146. INTERPOL

## 9.147. IRR

## 9.148. IS

## 9.149. JOIN

## 9.150. KEEP

## 9.151. LABEL

## 9.152. LAST

## 9.153. LASTOBS

## 9.154. LENGTH

## 9.155. LIB

## 9.156. LIBNAME

## 9.157. LIBRARY

## 9.158. LIKE

## 9.159. LISTING

## 9.160. LOGICAL

## 9.161. LONG

## 9.162. LOOKUP

## 9.163. LOOKUPCODE

## 9.164. LOOP

## 9.165. MAX

## 9.166. MEAN

## 9.167. MEANS

## 9.168. MEMORY

## 9.169. MERGE

## 9.170. MIN

## 9.171. MISSING

## 9.172. MOD

## 9.173. MULTIPLE

## 9.174. N

## 9.175. NAME

## 9.176. NBOBS

## 9.177. NEXT

## 9.178. NODATA

## 9.179. NODUPKEY

## 9.180. NOERROR

## 9.181. NOGRANDTOTAL

## 9.182. NOLIST

## 9.183. NOOBS

## 9.184. NORELOAD

## 9.185. NOT

## 9.186. NOTHING

## 9.187. NOTNULL

## 9.188. NULL

## 9.189. OBS

## 9.190. ODS

## 9.191. OFFICE

## 9.192. OR

## 9.193. OTHERWISE

## 9.194. OUT

## 9.195. OUTPUT

## 9.196. PARENT

## 9.197. PATH

## 9.198. PCTN

## 9.199. PCTSUM

## 9.200. PICTURE

## 9.201. PIE

## 9.202. PIE3D

## 9.203. PLOT

## 9.204. PREDATAROW

## 9.205. PREFIX

## 9.206. PRINT

## 9.207. PRINTCOL

## 9.208. PRINTFORM

## 9.209. PRIVATE

## 9.210. PROC

## 9.211. PUBLIC

## 9.212. PUT

## 9.213. REAL

## 9.214. RECURSE

## 9.215. RELATIONS

## 9.216. RENAME

## 9.217. REPORT\_DATE

## 9.218. RETURN

## 9.219. RUN

## 9.220. SELECT

## 9.221. SEMESTER

## 9.222. SEMESTERBEGDATE

## 9.223. SEMESTERENDDATE

## 9.224. SEMICOLON

## 9.225. SENDTOIC

## 9.226. SET

## 9.227. SETFILTER

## 9.228. SHARED

## 9.229. SHOWSELECTEDITEMS

## 9.230. SORT

## 9.231. SORTDESC

## 9.232. SPACE

## 9.233. SQL

## 9.234. SQLEXEC

## 9.235. SQLIMPORT

## 9.236. SQLTABLE

## 9.237. START

## 9.238. STOP

## 9.239. STRING

## 9.240. STYLE

## 9.241. STYLESHEET

## 9.242. SUBJECT

## 9.243. SUBSCRFUNDOP

## 9.244. SUBSCRFUNDOPSHARE

## 9.245. SUM

## 9.246. SUMVAR

## 9.247. SYMBOL

## 9.248. TAB

## 9.249. TABLE

## 9.250. TABLENOBS

## 9.251. TABLES

## 9.252. TABULATE

## 9.253. TEMPLATE

## 9.254. TEXT

## 9.255. THEN

## 9.256. TIME

## 9.257. TITLE

## 9.258. TITLE1

## 9.259. TITLE10

## 9.260. TITLE2

## 9.261. TITLE3

## 9.262. TITLE4

## 9.263. TITLE5

## 9.264. TITLE6

## 9.265. TITLE7

## 9.266. TITLE8

## 9.267. TITLE9

## 9.268. TO

## 9.269. TO

## 9.270. TOP

## 9.271. TRACE

## 9.272. TRANSACTIONS

## 9.273. TRANSPOSE

## 9.274. TRUE

## 9.275. TYPE

## 9.276. UNLOAD

## 9.277. UPCASE

## 9.278. UPDATE

## 9.279. URL

## 9.280. USE\_DRAFT

## 9.281. USE\_INVESTEE\_CURR

## 9.282. USE\_INVESTOR\_CURR

## 9.283. VALUE

## 9.284. VAR

## 9.285. VBAR

## 9.286. VBAR3D

## 9.287. VERTICALLY

## 9.288. WHEN

## 9.289. WHERE

## 9.290. WHILE

## 9.291. WIDTH

## 9.292. WITH