kenaflow Documentation

Release 4.0.23

kenaro GmbH

Nov 07, 2021

KENAFLOW

1 Workflows and Digitalization **3**

2 Editions **5**
2.1 *kenaflow* Standard . . . 5
2.2 *kenaflow* Developer . . . 5
2.3 *kenaflow* Free . . . 6

3 The Basics **7**

4 Architecture **9**
4.1 Enterprise . . . 9
4.1.1 Vertical Scaling = Instances . . . 9
4.1.2 Horizontal Scaling . . . 10
4.1.3 Vertical and Horizontal Scaling Together . . . 10

5 The Details **11**

6 New Features in 3.0 **15**
6.1 Multiple Webs and Lists . . . 15
6.2 Item Change Hash Fields . . . 16
6.3 Returning Data from Workflows using Remote Events . . . 16
6.4 Get an Answer from a Workflow using “Rich Links” . . . 16
6.5 Remote Event Debugging . . . 17

7 New Features in 4.0 **19**
7.1 Agents . . . 19
7.2 Performance . . . 19
7.3 Permanent Workflows . . . 19
7.4 Virtual Workflows . . . 19
7.5 PreProcessing Script and PostProcessing Script . . . 19
7.6 Pre-Defined Variables . . . 19
7.7 Storage Workflow . . . 20
7.8 Pre-Configured Execution Environment . . . 20
7.9 3 “Excel” related Cmdlets . . . 20

8 Setup **21**
8.1 Before You Start . . . 21
8.1.1 Prerequisites . . . 21
8.1.1.1 Prerequisites . . . 21
8.1.1.1.1 Hardware . . . 21
8.1.1.1.2 Operating System . . . 22
8.1.1.1.3 Server Side Storage . . . 22
8.1.1.1.4 Software . . . 23
8.1.2 Incompatible Software . . . 23
8.1.2.1 SharePoint PnP PowerShell . . . 23

8.1.2.2 Internet Information Server (IIS) . . . 24
8.1.3 System Changes . . . 24
8.1.3.1 Windows Registry . . . 24
8.1.3.2 Program Folder . . . 24
8.1.3.2.1 Program Data Folder . . . 24
8.1.3.3 http(s) Endpoints . . . 25
8.2 Installation . . . 25
8.2.1 Installation . . . 25
8.2.2 Complex Scenarios - Some Notes . . . 26
8.3 Basic Configuration . . . 26
8.3.1 SMTP Settings . . . 27
8.4 Before the First Start . . . 27
8.4.1 Attention . . . 27
8.4.2 Serial Code . . . 28
8.4.3 License Agreement . . . 28
8.4.4 Online Activation . . . 29
8.4.4.1 Multiple Instances of *kenaflow* . . . 29
8.4.5 Offline Activation . . . 29
8.4.6 Deactivation . . . 33
8.4.7 Service Account . . . 33
8.4.8 Automatic Execution . . . 34
8.4.9 Path Variable . . . 34

9 Operation **35**
9.1 Update . . . 35
9.1.1 Online update . . . 36
9.1.2 Offline update . . . 36
9.1.3 Typical Solutions . . . 37
9.1.4 Hints . . . 38
9.2 Logging . . . 39
9.2.1 Format . . . 40
9.2.1.1 Timestamp . . . 40
9.2.1.2 Process . . . 40
9.2.1.3 TID . . . 40
9.2.1.4 Area . . . 40
9.2.1.5 Category . . . 40
9.2.1.6 EventID . . . 40
9.2.1.7 Level . . . 41
9.2.1.8 Message . . . 41
9.2.1.9 Correlation . . . 41
9.2.2 Sample . . . 41
9.3 Monitoring . . . 53
9.3.1 Performance Counter . . . 53
9.3.2 Web Hooks . . . 53
9.3.3 Network Connectivity Check . . . 53
9.4 Installer Command Line . . . 54
9.4.1 Overview . . . 54
9.4.2 Setup . . . 54
9.4.3 Install Updates . . . 54
9.4.4 Update Automation . . . 55
9.5 Uninstall / Remove *kenaflow* . . . 55

10 Getting Started **57**
10.1 Workflow types . . . 60
10.2 Little video excursion . . . 61
10.3 What is a Workflow . . . 61
10.4 Workflow Processing . . . 62
10.5 Workflow Types . . . 63

10.5.1 State Machine Workflows . . . 63
10.5.1.1 Example . . . 63
10.5.1.2 Getting Started Knowledge . . . 63
10.5.1.3 Background Knowledge . . . 64
10.5.2 List Workflows . . . 66
10.5.2.1 Example . . . 66
10.5.2.2 Getting Started Knowledge . . . 66
10.5.2.3 Background Knowledge . . . 66
10.5.3 Site Workflows . . . 67
10.5.3.1 Example . . . 67
10.5.3.2 Getting Started Knowledge . . . 67
10.5.4 Email Workflows . . . 68
10.5.4.1 Example . . . 68
10.5.4.2 Getting Started Knowledge . . . 68
10.5.5 PowerShell Workflows . . . 68
10.5.5.1 Example . . . 68
10.5.5.2 Getting Started Knowledge . . . 69
10.6 Create workflows . . . 69
10.7 Structure of a Workflow . . . 70
10.8 Workflow Scripts . . . 70
10.8.1 Magic Line . . . 71
10.8.2 Other Workflow Scripts . . . 71
10.8.2.1 Pre-Query Script . . . 71
10.8.2.2 Error Handling Script . . . 72
10.8.2.2.1 Failure / Success . . . 73
10.8.3 Workflow Script Parameters . . . 73
10.8.3.1 State Machine Workflow . . . 74
10.8.3.2 List Workflow . . . 74
10.8.3.3 Site Workflow . . . 74
10.8.3.4 Email Workflow . . . 75
10.8.3.5 PowerShell Workflow . . . 75
10.9 Copy workflows . . . 75
10.9.1 Example . . . 75

11 Basics **77**
11.1 Templates . . . 78
11.1.1 String Processing (Invoke-KFTemplate) . . . 78
11.1.2 Special Values in the Config List Object . . . 79
11.1.3 Word / PowerPoint Processing . . . 80
11.2 Queries . . . 83
11.2.1 CAML . . . 83
11.2.1.1 Moderation State . . . 83
11.2.2 *kenaflow* Query Language . . . 84
11.3 Debugging . . . 85
11.3.1 Debugging of Workflow Scripts . . . 85
11.3.1.1 Error Handling Script Debugging . . . 86
11.3.1.2 Debugging of Pre-Query Scripts . . . 86
11.3.1.3 Preparing a debug session . . . 86
11.3.1.3.1 PowerShell Workflows . . . 86
11.3.1.3.2 SharePoint *Site* Workflows . . . 86
11.3.1.3.3 SharePoint *List* and *State Machine* Workflows . . . 87
11.3.1.3.4 Email Workflow with SharePoint Connection . . . 87
11.3.1.3.5 Email Workflow (without SharePoint Connection) . . . 87
11.3.1.4 Remote Event Debugging . . . 88
11.3.1.4.1 Debugging “Unstructured Data” Remove Event Receivers with Output Data . . . 91
11.4 Libraries . . . 93
11.5 Workflow Lists . . . 94

11.5.1 Config List . . . 94
11.5.2 Data List . . . 95
11.5.3 History List . . . 95
11.5.4 State List . . . 95
11.5.5 Error List . . . 96
11.6 Locking . . . 96
11.7 Links . . . 97
11.8 Workflow Safety . . . 98
11.8.1 Maximum Execution Time . . . 98
11.8.2 Script Run Timeout . . . 98
11.9 Version Upgrade . . . 98
11.9.1 Upgrade Paths . . . 99
11.9.2 1.0 => 2.0 . . . 99
11.10 Permissions . . . 99
11.10.1 SharePoint Permission Sets . . . 99
11.11 Remote Event . . . 101
11.11.1 Introduction . . . 101
11.11.2 The Execution of Remote Events . . . 101
11.11.3 Debugging of Remote Events . . . 102
11.11.4 SharePoint Remove Events . . . 102
11.11.5 Custom Data . . . 103
11.11.6 Unstructured Data . . . 104
11.11.7 Receiving Data from a Workflow . . . 105
11.11.8 Link Remote Events . . . 109
11.11.9 Behaviour in Case of Errors . . . 109
11.12 Error Handling . . . 110
11.12.1 Error Handling Basics . . . 110
11.12.2 Error Handling Scripts . . . 111
11.12.2.1 Failure / Success . . . 112
11.13 Persisting Data . . . 112
11.13.1 File System . . . 112
11.13.2 Database . . . 112
11.13.3 *kenaflow* Item Context Persistence . . . 112
11.13.4 *kenaflow* Data Persistence . . . 113
11.13.5 *kenaflow* SQLite Persistence . . . 113
11.14 Trigger . . . 113
11.14.1 Return Data . . . 114
11.15 The kenaflow Field . . . 115
11.15.1 General Information . . . 115
11.15.2 What is it for? . . . 115
11.15.3 Change Hash . . . 116
11.16 Storage . . . 116
11.16.1 Permanent Storage . . . 116
11.16.2 Session Storage . . . 116
11.17 Security Hardening . . . 117
11.17.1 Access to SharePoint . . . 117
11.17.1.1 Execution Account . . . 117
11.17.1.2 Access Account . . . 117
11.17.2 Access to Workflow Scripts / Workflow Configuration . . . 118
11.17.3 Instance Security . . . 118
11.18 Pre-Defined Variables . . . 119
11.18.1 Workflow Configurtation Parsing . . . 119
11.18.2 `PreWorkflowExecution` Script Block . . . 119
11.18.2.1 Additional Variables in EMail-IMAP Workflows . . . 120
11.18.2.2 Additional Variables in EMail-POP3 Workflows . . . 120
11.18.2.3 Additional Variables in EMail-Exchange Workflows . . . 120
11.18.2.4 Additional Variables in SharePoint Enabled EMail Workflows . . . 120
11.18.2.5 Additional in Storage Workflows . . . 120

11.18.2.6 Additional Variables in in (all) SharePoint Workflows . . . 120
11.18.2.7 Additional in SharePoint State Machine Workflows . . . 120
11.18.2.8 Additional Variables in PowerShell Workflows and Custom Workflows . . . 120
11.18.3 ErrorHandling-Script . . . 120
11.18.4 PreProcessing Script and PostProcessing Script . . . 121
11.18.5 (All) SharePoint Workflows . . . 121
11.18.5.1 Additional in SharePoint List Workflows and State Machine Workflows . . . 122
11.18.5.2 Additional Variables in SharePoint State Machine Workflows . . . 122
11.18.6 Pre Query Script of SharePoint List and State Machine Workflows . . . 122
11.18.7 Custom Workflow . . . 123
11.18.8 EMail Workflow . . . 123
11.18.8.1 Additional Variables in SharePoint Enabled EMail Workflows . . . 123

12 Advanced Configuration **125**
12.1 Storage . . . 125
12.2 Execution Slots . . . 125
12.3 Workflow Priority . . . 125
12.4 Storage . . . 125
12.5 Execution Slots . . . 126
12.6 Workflow Priority . . . 126
12.7 Advanced Workflow Configuration . . . 127

13 Agents **129**

14 Plugins **131**
14.1 Plugin Architecture Version **1** . . . 131
14.2 Plugin Architecture Version **2** . . . 131
14.3 Plugin Development . . . 131
14.3.1 Setup . . . 131
14.3.1.1 Setup `kenaflow.demoplugin.v1` . . . 132
14.3.1.2 Setup `kenaflow.demoplugin.v2` . . . 133
14.4 Plugin Architecture Version 1 . . . 133
14.4.1 Error handling . . . 148
14.5 kenaflow Cmdlets . . . 149
14.5.1 Basic Cmdlets . . . 149
14.5.2 Debugging and Runtime . . . 150
14.5.3 Workflow Control . . . 150
14.5.4 E-Mails, Templates, PDF . . . 151
14.5.5 Excel File Tools . . . 151
14.5.6 Tools . . . 151
14.5.7 Data Handling . . . 152
14.5.7.1 Available for all workflow types: . . . 152
14.5.7.2 Available only for SharePoint workflow types: . . . 153
14.5.7.3 Available only for SharePoint *list* and *state machine* workflow types: . . . 153
14.5.8 Locking . . . 153
14.5.8.1 SharePoint Specific Item Locking . . . 153
14.5.9 SharePoint Permissions, SharePoint Principals . . . 154
14.5.10 SharePoint Specials . . . 154
14.5.11 Add-KFPermission . . . 155
14.5.11.1 Syntax . . . 155
14.5.11.2 Support . . . 155
14.5.11.3 Return . . . 155
14.5.11.4 Description . . . 155
14.5.11.5 Parameters . . . 156
14.5.11.6 Examples . . . 157
14.5.11.7 Related Cmdlets . . . 157
14.5.12 Clear-KFDataTime . . . 158
14.5.12.1 Syntax . . . 158
14.5.12.2 Support . . . 158

14.5.12.3 Return . . . 158
14.5.12.4 Description . . . 158
14.5.12.5 Parameters . . . 158
14.5.12.6 Example . . . 158
14.5.12.7 Related Cmdlets . . . 158
14.5.13 ConvertFrom-KFISODateTime . . . 159
14.5.13.1 Syntax . . . 159
14.5.13.2 Support . . . 159
14.5.13.3 Return . . . 159
14.5.13.4 Description . . . 159
14.5.13.5 Parameters . . . 159
14.5.13.6 Example . . . 159
14.5.13.7 Related Cmdlets . . . 160
14.5.14 ConvertFrom-KFJSON . . . 160
14.5.14.1 Syntax . . . 160
14.5.14.2 Support . . . 160
14.5.14.3 Return . . . 160
14.5.14.4 Description . . . 161
14.5.14.5 Parameters . . . 161
14.5.14.6 Example . . . 161
14.5.15 ConvertoTo-KFCAML . . . 162
14.5.15.1 Syntax . . . 162
14.5.15.2 Support . . . 162
14.5.15.3 Return . . . 162
14.5.15.4 Description . . . 162
14.5.15.5 Parameters . . . 163
14.5.15.6 Example . . . 164
14.5.16 ConvertTo-KFISODateTime . . . 164
14.5.16.1 Syntax . . . 164
14.5.16.2 Support . . . 164
14.5.16.3 Return . . . 164
14.5.16.4 Description . . . 164
14.5.16.5 Parameters . . . 164
14.5.16.6 Example . . . 164
14.5.16.7 Related Cmdlets . . . 165
14.5.17 Get-KFBarcode . . . 165
14.5.17.1 Syntax . . . 165
14.5.17.2 Support . . . 166
14.5.17.3 Return . . . 166
14.5.17.4 Description . . . 166
14.5.17.5 Parameters . . . 166
14.5.17.6 Related Cmdlets . . . 166
14.5.18 Get-KFCron . . . 166
14.5.18.1 Syntax . . . 166
14.5.18.2 Support . . . 167
14.5.18.3 Return . . . 167
14.5.18.4 Description . . . 167
14.5.18.5 Parameters . . . 168
14.5.18.6 Example . . . 168
14.5.18.7 Related Cmdlets . . . 168
14.5.19 Get-KFCronUtc . . . 168
14.5.19.1 Syntax . . . 168
14.5.19.2 Support . . . 168
14.5.19.3 Return . . . 168
14.5.19.4 Description . . . 168
14.5.19.5 Parameters . . . 169
14.5.19.6 Example . . . 169
14.5.19.7 Related Cmdlets . . . 170

14.5.20 Get-KFData . . . 170
14.5.20.1 Syntax . . . 170
14.5.20.2 Support . . . 170
14.5.20.3 Return . . . 170
14.5.20.4 Description . . . 170
14.5.20.5 Parameters . . . 170
14.5.20.6 Example . . . 170
14.5.20.7 Related Cmdlets . . . 171
14.5.21 Get-KFDateTime . . . 171
14.5.21.1 Syntax . . . 171
14.5.21.2 Support . . . 171
14.5.21.3 Return . . . 171
14.5.21.4 Description . . . 171
14.5.21.5 Parameters . . . 172
14.5.21.6 Example . . . 173
14.5.21.7 Related Cmdlets . . . 173
14.5.22 Get-KFDBData . . . 173
14.5.22.1 Syntax . . . 173
14.5.22.2 Support . . . 173
14.5.22.3 Return . . . 173
14.5.22.4 Description . . . 173
14.5.22.5 Parameters . . . 174
14.5.22.6 Example . . . 174
14.5.22.7 Related Cmdlets . . . 174
14.5.23 Get-KFHash . . . 174
14.5.23.1 Syntax . . . 174
14.5.23.2 Support . . . 175
14.5.23.3 Return . . . 175
14.5.23.4 Description . . . 175
14.5.23.5 Parameters . . . 175
14.5.23.6 Example . . . 175
14.5.24 Get-KFInDebug . . . 175
14.5.24.1 Syntax . . . 175
14.5.24.2 Support . . . 176
14.5.24.3 Return . . . 176
14.5.24.4 Description . . . 176
14.5.24.5 Parameters . . . 176
14.5.24.6 Examples . . . 176
14.5.25 Get-KFItemData . . . 177
14.5.25.1 Syntax . . . 177
14.5.25.2 Support . . . 177
14.5.25.3 Return . . . 177
14.5.25.4 Description . . . 177
14.5.25.5 Parameters . . . 177
14.5.25.6 Example . . . 177
14.5.25.7 Related Cmdlets . . . 178
14.5.26 Get-KFLastResult . . . 178
14.5.26.1 Syntax . . . 178
14.5.26.2 Support . . . 178
14.5.26.3 Return . . . 178
14.5.26.4 Description . . . 178
14.5.26.5 Parameters . . . 178
14.5.26.6 Example . . . 179
14.5.27 Get-KFLink . . . 179
14.5.27.1 Syntax . . . 179
14.5.27.2 Support . . . 179
14.5.27.3 Return . . . 179
14.5.27.4 Description . . . 179

14.5.27.5 Parameters . . . 179
14.5.27.6 Example . . . 180
14.5.28 Get-KFNormalizedCAML . . . 180
14.5.28.1 Syntax . . . 180
14.5.28.2 Support . . . 180
14.5.28.3 Return . . . 180
14.5.28.4 Description . . . 180
14.5.28.5 Parameters . . . 181
14.5.28.6 Example . . . 181
14.5.29 Get-KFObject . . . 182
14.5.29.1 Syntax . . . 182
14.5.29.2 Support . . . 182
14.5.29.3 Return . . . 182
14.5.29.4 Description . . . 182
14.5.29.5 Parameters . . . 182
14.5.29.6 Example . . . 182
14.5.29.7 Related Cmdlets . . . 183
14.5.30 Get-KFPermission . . . 183
14.5.30.1 Syntax . . . 183
14.5.30.2 Support . . . 183
14.5.30.3 Return . . . 183
14.5.30.4 Description . . . 183
14.5.30.5 Parameters . . . 183
14.5.30.6 Examples . . . 183
14.5.30.7 Related Cmdlets . . . 184
14.5.31 Get-KFPrettyXML . . . 184
14.5.31.1 Syntax . . . 184
14.5.31.2 Support . . . 184
14.5.31.3 Return . . . 184
14.5.31.4 Description . . . 184
14.5.31.5 Parameters . . . 184
14.5.31.6 Example . . . 184
14.5.32 Get-KFScriptRunTimeout . . . 185
14.5.32.1 Syntax . . . 185
14.5.32.2 Support . . . 186
14.5.32.3 Description . . . 186
14.5.32.4 Parameter . . . 186
14.5.32.5 Examples . . . 186
14.5.33 Get-KFSessionStorageData . . . 186
14.5.33.1 Syntax . . . 186
14.5.33.2 Support . . . 186
14.5.33.3 Return . . . 187
14.5.33.4 Description . . . 187
14.5.33.5 Parameters . . . 187
14.5.33.6 Example . . . 188
14.5.33.7 Related Cmdlets . . . 188
14.5.34 Get-KFSPDatetimeLocal . . . 188
14.5.34.1 Syntax . . . 188
14.5.34.2 Support . . . 188
14.5.34.3 Return . . . 188
14.5.34.4 Description . . . 188
14.5.34.5 Parameters . . . 189
14.5.34.6 Example . . . 189
14.5.34.7 Related Cmdlets . . . 189
14.5.35 Get-KFStorageData . . . 189
14.5.35.1 Syntax . . . 189
14.5.35.2 Support . . . 190
14.5.35.3 Return . . . 190

14.5.35.4 Description . . . 190
14.5.35.5 Parameters . . . 190
14.5.35.6 Example . . . 191
14.5.35.7 Related Cmdlets . . . 191
14.5.36 Get-KFTest . . . 191
14.5.36.1 Syntax . . . 191
14.5.36.2 Support . . . 191
14.5.36.3 Return . . . 191
14.5.36.4 Description . . . 191
14.5.36.5 Parameters . . . 191
14.5.37 Get-KFThrowErrorsPreference . . . 192
14.5.37.1 Syntax . . . 192
14.5.37.2 Support . . . 192
14.5.37.3 Return . . . 192
14.5.37.4 Description . . . 192
14.5.37.5 Parameters . . . 192
14.5.37.6 Example . . . 192
14.5.37.7 Related Cmdlets . . . 192
14.5.38 Get-KFUrl . . . 192
14.5.38.1 Syntax . . . 192
14.5.38.2 Support . . . 193
14.5.38.3 Return . . . 194
14.5.38.4 Description . . . 194
14.5.38.5 Parameters . . . 194
14.5.38.6 Example . . . 195
14.5.39 Get-KFUserInfo . . . 195
14.5.39.1 Syntax . . . 195
14.5.39.2 Support . . . 195
14.5.39.3 Return . . . 195
14.5.39.4 Description . . . 195
14.5.39.5 Parameters . . . 197
14.5.39.6 Example . . . 198
14.5.39.7 Related Cmdlets . . . 198
14.5.40 Get-KFUsers . . . 198
14.5.40.1 Syntax . . . 198
14.5.40.2 Description . . . 198
14.5.41 Import-KFExcelFile . . . 198
14.5.41.1 Syntax . . . 198
14.5.41.2 Support . . . 198
14.5.41.3 Return . . . 199
14.5.41.4 Description . . . 199
14.5.41.5 Parameters . . . 199
14.5.41.6 Example . . . 200
14.5.41.6.1 1. . . . 200
14.5.41.6.2 2. . . . 201
14.5.41.7 Related Cmdlets . . . 202
14.5.42 Invoke-Kenaflow . . . 202
14.5.42.1 Syntax . . . 202
14.5.42.2 Support . . . 202
14.5.42.3 Return . . . 203
14.5.42.4 Description . . . 203
14.5.42.5 Parameters . . . 204
14.5.42.6 Example . . . 205
14.5.43 Invoke-KFCtrlC . . . 205
14.5.43.1 Syntax . . . 205
14.5.43.2 Support . . . 205
14.5.43.3 Return . . . 205
14.5.43.4 Description . . . 205

14.5.43.5 Parameters . . . 205
14.5.43.6 Example . . . 205
14.5.43.7 Related Cmdlets . . . 205
14.5.44 Invoke-KFCustomDataEventSimulation . . . 206
14.5.44.1 Syntax . . . 206
14.5.44.2 Support . . . 206
14.5.44.3 Return . . . 206
14.5.44.4 Description . . . 206
14.5.44.5 Parameters . . . 207
14.5.44.6 Example . . . 207
14.5.44.7 Related Cmdlets . . . 207
14.5.45 Invoke-KFDebugBreak . . . 208
14.5.45.1 Syntax . . . 208
14.5.45.2 Support . . . 208
14.5.45.3 Return . . . 208
14.5.45.4 Description . . . 208
14.5.45.5 Parameters . . . 208
14.5.45.6 Example . . . 208
14.5.46 Invoke-KFException . . . 208
14.5.46.1 Syntax . . . 208
14.5.46.2 Support . . . 208
14.5.46.3 Return . . . 209
14.5.46.4 Description . . . 209
14.5.46.5 Parameters . . . 209
14.5.46.6 Example . . . 209
14.5.46.7 Related Cmdlets . . . 209
14.5.47 Invoke-KFFileTemplate . . . 209
14.5.47.1 Syntax . . . 209
14.5.47.2 Support . . . 210
14.5.47.3 Return . . . 210
14.5.47.4 Description . . . 210
14.5.47.5 Parameters . . . 211
14.5.47.6 Examples . . . 212
14.5.47.7 Related Cmdlets . . . 212
14.5.48 Invoke-KFGlobalStop . . . 212
14.5.48.1 Syntax . . . 212
14.5.48.2 Support . . . 212
14.5.48.3 Return . . . 212
14.5.48.4 Description . . . 212
14.5.48.5 Parameters . . . 213
14.5.48.6 Example . . . 213
14.5.48.7 Related Cmdlets . . . 213
14.5.49 Invoke-KFLinkEventSimulation . . . 213
14.5.49.1 Syntax . . . 213
14.5.49.2 Support . . . 213
14.5.49.3 Return . . . 213
14.5.49.4 Description . . . 214
14.5.49.5 Parameters . . . 214
14.5.49.6 Example . . . 215
14.5.49.7 Related Cmdlets . . . 215
14.5.50 Invoke-KFPipelineStop . . . 215
14.5.50.1 Syntax . . . 215
14.5.50.2 Support . . . 215
14.5.50.3 Return . . . 215
14.5.50.4 Description . . . 215
14.5.50.5 Parameters . . . 215
14.5.50.6 Example . . . 216
14.5.50.7 Related Cmdlets . . . 216

14.5.51 Invoke-KFSharePointEventSimulation . . . 216
14.5.51.1 Syntax . . . 216
14.5.51.2 Support . . . 216
14.5.51.3 Return . . . 216
14.5.51.4 Description . . . 216
14.5.51.5 Parameters . . . 217
14.5.51.6 Example . . . 217
14.5.51.7 Related Cmdlets . . . 218
14.5.52 Invoke-KFTemplate . . . 218
14.5.52.1 Syntax . . . 218
14.5.52.2 Support . . . 218
14.5.52.3 Return . . . 218
14.5.52.4 Description . . . 218
14.5.52.5 Parameters . . . 218
14.5.52.6 Examples . . . 219
14.5.52.7 Related Cmdlets . . . 219
14.5.53 Invoke-KFTerminatePosh . . . 219
14.5.53.1 Syntax . . . 219
14.5.53.2 Support . . . 219
14.5.53.3 Return . . . 219
14.5.53.4 Description . . . 219
14.5.53.5 Parameters . . . 220
14.5.53.6 Example . . . 220
14.5.53.7 Related Cmdlets . . . 220
14.5.54 Invoke-KFUnstructuredDataEventSimulation . . . 220
14.5.54.1 Syntax . . . 220
14.5.54.2 Support . . . 220
14.5.54.3 Return . . . 220
14.5.54.4 Description . . . 221
14.5.54.5 Parameters . . . 221
14.5.54.6 Example . . . 222
14.5.54.6.1 Example “Get-PDF” . . . 222
14.5.54.7 Related Cmdlets . . . 223
14.5.55 Invoke-KFWorkflow . . . 223
14.5.55.1 Syntax . . . 223
14.5.55.2 Support . . . 224
14.5.55.3 Return . . . 224
14.5.55.4 Description . . . 224
14.5.55.5 Parameters . . . 224
14.5.55.6 Example . . . 225
14.5.56 Lock-KFItem . . . 225
14.5.56.1 Syntax . . . 225
14.5.56.2 Support . . . 225
14.5.56.3 Return . . . 225
14.5.56.4 Description . . . 225
14.5.56.5 Parameters . . . 225
14.5.56.6 Example . . . 226
14.5.57 Lock-KFProcess . . . 226
14.5.57.1 Syntax . . . 226
14.5.57.2 Support . . . 226
14.5.57.3 Return . . . 226
14.5.57.4 Description . . . 226
14.5.57.5 Parameters . . . 226
14.5.57.6 Example . . . 227
14.5.57.7 Related Cmdlets . . . 227
14.5.58 New-KFBarcode . . . 227
14.5.58.1 Syntax . . . 227
14.5.58.2 Support . . . 228

14.5.58.3 Return . . . 228
14.5.58.4 Description . . . 228
14.5.58.5 Parameters . . . 229
14.5.58.6 Example . . . 230
14.5.58.7 Related Cmdlets . . . 231
14.5.59 New-KFExcelFile . . . 231
14.5.59.1 Syntax . . . 231
14.5.59.2 Support . . . 231
14.5.59.3 Return . . . 231
14.5.59.4 Description . . . 231
14.5.59.5 Parameters . . . 232
14.5.59.6 Example . . . 233
14.5.59.7 Related Cmdlets . . . 236
14.5.60 New-KFPDF . . . 236
14.5.60.1 Syntax . . . 236
14.5.60.2 Support . . . 236
14.5.60.3 Return . . . 237
14.5.60.4 Description . . . 237
14.5.60.5 Parameters . . . 237
14.5.60.6 Example . . . 237
14.5.60.7 Related Cmdlets . . . 237
14.5.61 Protect-KFData . . . 237
14.5.61.1 Syntax . . . 237
14.5.61.2 Support . . . 238
14.5.61.3 Return . . . 238
14.5.61.4 Description . . . 238
14.5.61.5 Parameters . . . 238
14.5.61.6 Examples . . . 238
14.5.61.7 Related Cmdlets . . . 238
14.5.62 Remove-KFData . . . 238
14.5.62.1 Syntax . . . 238
14.5.62.2 Support . . . 239
14.5.62.3 Return . . . 239
14.5.62.4 Description . . . 239
14.5.62.5 Parameters . . . 239
14.5.62.6 Example . . . 239
14.5.62.7 Related Cmdlets . . . 239
14.5.63 Remove-KFDBData . . . 239
14.5.63.1 Syntax . . . 239
14.5.63.2 Support . . . 240
14.5.63.3 Return . . . 240
14.5.63.4 Description . . . 240
14.5.63.5 Parameters . . . 240
14.5.63.6 Example . . . 240
14.5.63.7 Related Cmdlets . . . 240
14.5.64 Remove-KFItem . . . 241
14.5.64.1 Syntax . . . 241
14.5.64.2 Support . . . 241
14.5.64.3 Return . . . 241
14.5.64.4 Description . . . 241
14.5.64.5 Parameters . . . 241
14.5.64.6 Example . . . 241
14.5.65 Remove-KFItemData . . . 241
14.5.65.1 Syntax . . . 241
14.5.65.2 Support . . . 242
14.5.65.3 Return . . . 242
14.5.65.4 Description . . . 242
14.5.65.5 Parameters . . . 242

14.5.65.6 Example . . . 242
14.5.65.7 Related Cmdlets . . . 242
14.5.66 Remove-KFObject . . . 242
14.5.66.1 Syntax . . . 242
14.5.66.2 Support . . . 243
14.5.66.3 Return . . . 243
14.5.66.4 Description . . . 243
14.5.66.5 Parameters . . . 243
14.5.66.6 Example . . . 243
14.5.66.7 Related Cmdlets . . . 244
14.5.67 Remove-KFPermission . . . 244
14.5.67.1 Syntax . . . 244
14.5.67.2 Support . . . 244
14.5.67.3 Return . . . 244
14.5.67.4 Description . . . 244
14.5.67.5 Parameters . . . 245
14.5.67.6 Examples . . . 246
14.5.67.7 Related Cmdlets . . . 246
14.5.68 Remove-KFSessionStorageData . . . 247
14.5.68.1 Syntax . . . 247
14.5.68.2 Support . . . 247
14.5.68.3 Return . . . 247
14.5.68.4 Description . . . 247
14.5.68.5 Parameters . . . 247
14.5.68.6 Example . . . 248
14.5.68.7 Related Cmdlets . . . 248
14.5.69 Remove-KFStorageData . . . 248
14.5.69.1 Syntax . . . 248
14.5.69.2 Support . . . 248
14.5.69.3 Return . . . 248
14.5.69.4 Description . . . 248
14.5.69.5 Parameters . . . 249
14.5.69.6 Example . . . 249
14.5.69.7 Related Cmdlets . . . 249
14.5.70 Reset-KFPermissionInheritance . . . 249
14.5.70.1 Syntax . . . 249
14.5.70.2 Support . . . 249
14.5.70.3 Return . . . 250
14.5.70.4 Description . . . 250
14.5.70.5 Parameters . . . 250
14.5.70.6 Example . . . 250
14.5.70.7 Related Cmdlets . . . 250
14.5.71 Resolve-KFPrincipal . . . 250
14.5.71.1 Syntax . . . 250
14.5.71.2 Support . . . 251
14.5.71.3 Return . . . 251
14.5.71.4 Description . . . 251
14.5.71.5 Parameters . . . 253
14.5.71.6 Example . . . 253
14.5.72 Send-KFMail . . . 253
14.5.72.1 Syntax . . . 253
14.5.72.2 Support . . . 254
14.5.72.3 Return . . . 254
14.5.72.4 Description . . . 254
14.5.72.4.1 Attachments . . . 255
14.5.72.4.2 Best Practice Pattern . . . 256
14.5.72.5 Parameter . . . 256
14.5.72.6 Examples . . . 257

14.5.72.7 Related Cmdlets . . . 258
14.5.73 Set-KFApproval . . . 258
14.5.73.1 Syntax . . . 258
14.5.73.2 Support . . . 258
14.5.73.3 Return . . . 258
14.5.73.4 Description . . . 259
14.5.73.5 Parameters . . . 259
14.5.73.6 Example . . . 259
14.5.73.7 Related Cmdlets . . . 259
14.5.74 Set-KFContext . . . 260
14.5.75 Set-KFData . . . 260
14.5.75.1 Syntax . . . 260
14.5.75.2 Support . . . 260
14.5.75.3 Return . . . 260
14.5.75.4 Description . . . 260
14.5.75.5 Parameters . . . 260
14.5.75.6 Example . . . 261
14.5.75.7 Related Cmdlets . . . 261
14.5.76 Set-KFDateTime . . . 261
14.5.76.1 Syntax . . . 261
14.5.76.2 Support . . . 261
14.5.76.3 Return . . . 261
14.5.76.4 Description . . . 261
14.5.76.5 Parameters . . . 262
14.5.76.6 Example . . . 262
14.5.76.7 Related Cmdlets . . . 262
14.5.77 Set-KFDBData . . . 262
14.5.77.1 Syntax . . . 262
14.5.77.2 Support . . . 262
14.5.77.3 Return . . . 262
14.5.77.4 Description . . . 262
14.5.77.5 Parameters . . . 263
14.5.77.6 Example . . . 263
14.5.77.7 Related Cmdlets . . . 263
14.5.78 Set-KFFinalAuthor . . . 263
14.5.78.1 Syntax . . . 263
14.5.78.2 Support . . . 263
14.5.78.3 Return . . . 264
14.5.78.4 Description . . . 264
14.5.78.5 Parameters . . . 264
14.5.78.6 Example . . . 264
14.5.78.7 Related Cmdlets . . . 264
14.5.79 Set-KFFinalCheckin . . . 264
14.5.79.1 Syntax . . . 264
14.5.79.2 Support . . . 264
14.5.79.3 Return . . . 265
14.5.79.4 Description . . . 265
14.5.79.5 Parameters . . . 265
14.5.79.6 Example . . . 265
14.5.79.7 Related Cmdlets . . . 265
14.5.80 Set-KFFinalEditor . . . 266
14.5.80.1 Syntax . . . 266
14.5.80.2 Support . . . 266
14.5.80.3 Return . . . 266
14.5.80.4 Description . . . 266
14.5.80.5 Parameters . . . 266
14.5.80.6 Example . . . 266
14.5.80.7 Related Cmdlets . . . 267

14.5.81 Set-KFFinalModerationState . . . 267
14.5.81.1 Syntax . . . 267
14.5.81.2 Support . . . 267
14.5.81.3 Return . . . 267
14.5.81.4 Description . . . 267
14.5.81.5 Parameters . . . 267
14.5.81.6 Example . . . 268
14.5.81.7 Related Cmdlets . . . 268
14.5.82 Set-KFItemData . . . 268
14.5.82.1 Syntax . . . 268
14.5.82.2 Support . . . 268
14.5.82.3 Return . . . 268
14.5.82.4 Description . . . 268
14.5.82.5 Parameters . . . 269
14.5.82.6 Example . . . 269
14.5.82.7 Related Cmdlets . . . 269
14.5.83 Set-KFItemModifiedFlag . . . 269
14.5.83.1 Syntax . . . 269
14.5.83.2 Support . . . 269
14.5.83.3 Return . . . 269
14.5.83.4 Description . . . 270
14.5.83.5 Parameters . . . 270
14.5.83.6 Example . . . 270
14.5.83.7 Related Cmdlets . . . 270
14.5.84 Set-KFObject . . . 270
14.5.84.1 Syntax . . . 270
14.5.84.2 Support . . . 270
14.5.84.3 Return . . . 270
14.5.84.4 Description . . . 270
14.5.84.5 Parameters . . . 271
14.5.84.6 Example . . . 271
14.5.84.7 Related Cmdlets . . . 271
14.5.85 Set-KFPermission . . . 271
14.5.85.1 Syntax . . . 271
14.5.85.2 Support . . . 272
14.5.85.3 Return . . . 272
14.5.85.4 Description . . . 272
14.5.85.5 Parameters . . . 273
14.5.85.6 Examples . . . 274
14.5.85.7 Related Cmdlets . . . 274
14.5.86 Set-KFScriptRunTimeout . . . 274
14.5.86.1 Syntax . . . 274
14.5.86.2 Support . . . 274
14.5.86.3 Description . . . 275
14.5.86.4 Parameter . . . 275
14.5.86.5 Examples . . . 275
14.5.87 Set-KFSessionStorageData . . . 275
14.5.87.1 Syntax . . . 275
14.5.87.2 Support . . . 276
14.5.87.3 Return . . . 276
14.5.87.4 Description . . . 276
14.5.87.5 Parameters . . . 276
14.5.87.6 Example . . . 277
14.5.87.7 Related Cmdlets . . . 277
14.5.88 Set-KFSPDatetimeLocal . . . 277
14.5.88.1 Syntax . . . 277
14.5.88.2 Support . . . 277
14.5.88.3 Return . . . 277

14.5.88.4 Description . . . 277
14.5.88.5 Parameters . . . 277
14.5.88.6 Example . . . 278
14.5.88.7 Related Cmdlets . . . 278
14.5.89 Set-KFState . . . 278
14.5.89.1 Syntax . . . 278
14.5.89.2 Support . . . 278
14.5.89.3 Return . . . 278
14.5.89.4 Description . . . 278
14.5.89.5 Parameters . . . 279
14.5.89.6 Example . . . 279
14.5.90 Set-KFStorageData . . . 280
14.5.90.1 Syntax . . . 280
14.5.90.2 Support . . . 280
14.5.90.3 Return . . . 280
14.5.90.4 Description . . . 280
14.5.90.5 Parameters . . . 280
14.5.90.6 Example . . . 281
14.5.90.7 Related Cmdlets . . . 281
14.5.91 Set-KFThrowErrorsPreference . . . 281
14.5.91.1 Syntax . . . 281
14.5.91.2 Support . . . 281
14.5.91.3 Return . . . 281
14.5.91.4 Description . . . 282
14.5.91.5 Parameters . . . 282
14.5.91.6 Example . . . 282
14.5.91.7 Related Cmdlets . . . 282
14.5.92 Start-KFSharePointModule . . . 282
14.5.92.1 Syntax . . . 282
14.5.92.2 Support . . . 282
14.5.92.3 Return . . . 282
14.5.92.4 Description . . . 282
14.5.92.5 Parameters . . . 283
14.5.92.6 Example . . . 283
14.5.93 Stop-KFPermissionInheritance . . . 283
14.5.93.1 Syntax . . . 283
14.5.93.2 Support . . . 283
14.5.93.3 Return . . . 283
14.5.93.4 Description . . . 283
14.5.93.5 Parameters . . . 284
14.5.93.6 Example . . . 284
14.5.93.7 Related Cmdlets . . . 284
14.5.94 Stop-KFScript . . . 284
14.5.94.1 Syntax . . . 284
14.5.94.2 Support . . . 285
14.5.94.3 Return . . . 285
14.5.94.4 Description . . . 285
14.5.94.5 Parameters . . . 285
14.5.94.6 Examples . . . 286
14.5.95 Test-KFConfig . . . 286
14.5.95.1 Syntax . . . 286
14.5.95.2 Support . . . 286
14.5.95.3 Return . . . 286
14.5.95.4 Description . . . 286
14.5.95.5 Parameters . . . 286
14.5.96 Test-KFItemLock . . . 286
14.5.96.1 Syntax . . . 286
14.5.96.2 Support . . . 287

14.5.96.3 Return . . . 287
14.5.96.4 Description . . . 287
14.5.96.5 Parameters . . . 287
14.5.96.6 Example . . . 287
14.5.96.7 Related Cmdlets . . . 287
14.5.97 Test-KFProcessLock . . . 287
14.5.97.1 Syntax . . . 287
14.5.97.2 Support . . . 288
14.5.97.3 Return . . . 288
14.5.97.4 Description . . . 288
14.5.97.5 Parameters . . . 288
14.5.97.6 Example . . . 288
14.5.97.7 Related Cmdlets . . . 288
14.5.98 Unlock-KFItem . . . 288
14.5.98.1 Syntax . . . 288
14.5.98.2 Support . . . 289
14.5.98.3 Return . . . 289
14.5.98.4 Description . . . 289
14.5.98.5 Parameters . . . 289
14.5.98.6 Example . . . 289
14.5.98.7 Related Cmdlets . . . 289
14.5.99 Unlock-KFProcess . . . 290
14.5.99.1 Syntax . . . 290
14.5.99.2 Support . . . 290
14.5.99.3 Return . . . 290
14.5.99.4 Description . . . 290
14.5.99.5 Parameters . . . 290
14.5.99.6 Example . . . 290
14.5.99.7 Related Cmdlets . . . 291
14.5.100 Unprotect-KFData . . . 291
14.5.100.1 Syntax . . . 291
14.5.100.2 Support . . . 291
14.5.100.3 Return . . . 291
14.5.100.4 Description . . . 291
14.5.100.5 Parameters . . . 291
14.5.100.6 Example . . . 292
14.5.100.7 Related Cmdlets . . . 292
14.5.101 Update-KFExcelFile . . . 292
14.5.101.1 Syntax . . . 292
14.5.101.2 Support . . . 292
14.5.101.3 Return . . . 293
14.5.101.4 Description . . . 293
14.5.101.5 Parameters . . . 293
14.5.101.6 Example . . . 294
14.5.101.7 Related Cmdlets . . . 296
14.5.102 Update-KFPermission . . . 296
14.5.102.1 Syntax . . . 296
14.5.102.2 Support . . . 296
14.5.102.3 Return . . . 296
14.5.102.4 Description . . . 296
14.5.102.5 Parameters . . . 298
14.5.102.6 Examples . . . 298
14.5.102.7 Related Cmdlets . . . 299
14.5.103 Write-KFHistory . . . 299
14.5.103.1 Syntax . . . 299
14.5.103.2 Support . . . 299
14.5.103.3 Return . . . 299
14.5.103.4 Description . . . 299

14.5.103.5 Parameters . . . 299
14.5.103.6 Examples . . . 299
14.5.103.7 Related Cmdlets . . . 300
14.5.104 Write-KFLog . . . 300
14.5.104.1 Syntax . . . 300
14.5.104.2 Support . . . 300
14.5.104.3 Return . . . 300
14.5.104.4 Description . . . 300
14.5.104.5 Parameters . . . 300
14.5.104.6 Example . . . 301
14.5.104.7 Related Cmdlets . . . 301
14.5.105 Write-KFLogOutput . . . 301
14.5.105.1 Syntax . . . 301
14.5.105.2 Support . . . 301
14.5.105.3 Return . . . 301
14.5.105.4 Description . . . 301
14.5.105.5 Parameters . . . 301
14.5.105.6 Example . . . 302
14.5.105.7 Related Cmdlets . . . 302

15 Command Line Reference **303**
15.1 Runtime . . . 303
15.1.1 –run . . . 304
15.1.1.1 Syntax . . . 304
15.1.1.2 Description . . . 304
15.1.1.2.1 Examples . . . 304
15.1.2 –exec . . . 305
15.1.2.1 Syntax . . . 305
15.1.2.2 Description . . . 305
15.1.3 –rer . . . 305
15.1.3.1 Syntax . . . 305
15.1.3.2 Description . . . 305
15.1.4 –norer . . . 306
15.1.4.1 Syntax . . . 306
15.1.4.2 Description . . . 306
15.1.4.3 Related switches . . . 306
15.1.5 –norerspo . . . 306
15.1.5.1 Syntax . . . 306
15.1.5.2 Description . . . 306
15.1.5.3 Related switches . . . 306
15.1.6 –norersp2013 . . . 306
15.1.6.1 Syntax . . . 306
15.1.6.2 Description . . . 307
15.1.6.3 Related switches . . . 307
15.1.7 –norersp2016 . . . 307
15.1.7.1 Syntax . . . 307
15.1.7.2 Description . . . 307
15.1.7.3 Related switches . . . 307
15.1.8 –norersp2019 . . . 307
15.1.8.1 Syntax . . . 307
15.1.8.2 Description . . . 308
15.1.8.3 Related switches . . . 308
15.1.9 –norerposh . . . 308
15.1.9.1 Syntax . . . 308
15.1.9.2 Description . . . 308
15.1.9.3 Related switches . . . 308
15.1.10 –rerrunnersp2013 . . . 308
15.1.10.1 Syntax . . . 308

15.1.10.2 Description . . . 309
15.1.10.3 Related switches . . . 309
15.1.11 –rerrunnersp2016 . . . 309
15.1.11.1 Syntax . . . 309
15.1.11.2 Description . . . 309
15.1.11.3 Related switches . . . 309
15.1.12 –rerrunnersp2019 . . . 309
15.1.12.1 Syntax . . . 309
15.1.12.2 Description . . . 310
15.1.12.3 Related switches . . . 310
15.1.13 –rerrunnerspo . . . 310
15.1.13.1 Syntax . . . 310
15.1.13.2 Description . . . 310
15.1.13.3 Related switches . . . 310
15.1.14 –rerrunnerposh . . . 310
15.1.14.1 Syntax . . . 310
15.1.14.2 Description . . . 311
15.1.14.3 Related switches . . . 311
15.2 Setup and Operation . . . 311
15.2.1 –enable . . . 311
15.2.1.1 Syntax . . . 311
15.2.1.2 Description . . . 311
15.2.2 –disable . . . 312
15.2.2.1 Syntax . . . 312
15.2.2.2 Description . . . 312
15.2.3 –editconfig . . . 312
15.2.3.1 Syntax . . . 312
15.2.3.2 Description . . . 312
15.2.4 –changeconfigsetting . . . 313
15.2.4.1 Syntax . . . 313
15.2.4.2 Description . . . 313
15.2.5 –securing . . . 313
15.2.5.1 Syntax . . . 313
15.2.5.2 Description . . . 313
15.2.5.3 Example . . . 314
15.2.6 –installschedtask . . . 314
15.2.6.1 Syntax . . . 314
15.2.6.2 Description . . . 314
15.2.7 –createperformancecounter . . . 314
15.2.7.1 Syntax . . . 314
15.2.7.2 Description . . . 314
15.2.8 –removeperformancecounter . . . 314
15.2.8.1 Syntax . . . 314
15.2.8.2 Description . . . 315
15.2.8.3 Example . . . 315
15.2.9 –testperformancecounter . . . 315
15.2.9.1 Syntax . . . 315
15.2.9.2 Description . . . 315
15.2.10 –decryptconfig . . . 315
15.2.10.1 Syntax . . . 315
15.2.10.2 Description . . . 315
15.2.11 –configthttp . . . 315
15.2.11.1 Syntax . . . 315
15.2.11.2 Description . . . 316
15.2.11.3 Example . . . 316
15.2.12 –sendtestmail . . . 316
15.2.12.1 Syntax . . . 316
15.2.12.2 Description . . . 317

15.2.13 –updatenetsecurity . . . 317
15.2.13.1 Syntax . . . 317
15.2.13.2 Description . . . 317
15.2.13.3 Examples . . . 317
15.2.14 –reset . . . 317
15.2.14.1 Syntax . . . 317
15.2.14.2 Description . . . 318
15.2.14.3 Examples . . . 318
15.3 Workflow Management . . . 318
15.3.1 Remote Events . . . 318
15.3.1.1 Management . . . 318
15.3.1.2 Initialization . . . 318
15.3.2 –registerrer . . . 319
15.3.2.1 Syntax . . . 319
15.3.2.2 Description . . . 319
15.3.3 –unregisterrer . . . 319
15.3.3.1 Syntax . . . 319
15.3.3.2 Description . . . 320
15.3.4 –new . . . 320
15.3.4.1 Syntax . . . 320
15.3.4.2 Description . . . 320
15.3.5 –setusername . . . 320
15.3.5.1 Syntax . . . 320
15.3.5.2 Description . . . 321
15.3.6 –appid . . . 321
15.3.6.1 Syntax . . . 321
15.3.6.2 Description . . . 321
15.3.6.3 Syntax . . . 321
15.3.6.4 Description . . . 321
15.3.7 –removecomments . . . 321
15.3.7.1 Syntax . . . 321
15.3.7.2 Description . . . 321
15.3.7.3 Related Switches . . . 322
15.3.8 –backupconfiglist . . . 322
15.3.8.1 Syntax . . . 322
15.3.8.2 Description . . . 322
15.3.8.3 Example . . . 323
15.3.9 –copy . . . 323
15.3.9.1 Syntax . . . 323
15.3.9.2 Description . . . 323
15.3.10 –createpermissionsets . . . 323
15.3.10.1 Syntax . . . 323
15.3.10.2 Description . . . 323
15.3.10.3 Related Switches . . . 324
15.3.11 –createconfiglist . . . 324
15.3.11.1 Syntax . . . 324
15.3.11.2 Description . . . 324
15.3.11.3 Related Switches . . . 325
15.3.12 –createstatelist . . . 325
15.3.12.1 Syntax . . . 325
15.3.12.2 Description . . . 325
15.3.12.3 Related Switches . . . 326
15.3.13 –createerrorlist . . . 326
15.3.13.1 Syntax . . . 326
15.3.13.2 Description . . . 326
15.3.13.3 Related Switches . . . 327
15.3.14 –createdatalist . . . 327
15.3.14.1 Syntax . . . 327

15.3.14.2 Description . . . 327
15.3.14.3 Related Switches . . . 328
15.3.15 –createhistorylist . . . 328
15.3.15.1 Syntax . . . 328
15.3.15.2 Description . . . 328
15.3.15.3 Related Switches . . . 329
15.3.16 –showstatelist . . . 329
15.3.16.1 Syntax . . . 329
15.3.16.2 Description . . . 329
15.3.16.3 Related Switches . . . 330
15.3.17 –showerrorlist . . . 330
15.3.17.1 Syntax . . . 330
15.3.17.2 Description . . . 330
15.3.17.3 Related Switches . . . 331
15.3.18 –showdatalist . . . 331
15.3.18.1 Syntax . . . 331
15.3.18.2 Description . . . 331
15.3.18.3 Related Switches . . . 332
15.3.19 –showhistorylist . . . 332
15.3.19.1 Syntax . . . 332
15.3.19.2 Description . . . 332
15.3.19.3 Related Switches . . . 333
15.3.20 –hidestatelist . . . 333
15.3.20.1 Syntax . . . 333
15.3.20.2 Description . . . 333
15.3.20.3 Related Switches . . . 334
15.3.21 –hideerrorlist . . . 334
15.3.21.1 Syntax . . . 334
15.3.21.2 Description . . . 334
15.3.21.3 Related Switches . . . 335
15.3.22 –hidedatalist . . . 335
15.3.22.1 Syntax . . . 335
15.3.22.2 Description . . . 335
15.3.22.3 Related Switches . . . 336
15.3.23 –hidehistorylist . . . 336
15.3.23.1 Syntax . . . 336
15.3.23.2 Description . . . 336
15.3.23.3 Related Switches . . . 337
15.3.24 –showconfiglist . . . 337
15.3.24.1 Syntax . . . 337
15.3.24.2 Description . . . 337
15.3.24.3 Related Switches . . . 338
15.3.25 –hideconfiglist . . . 338
15.3.25.1 Syntax . . . 338
15.3.25.2 Description . . . 338
15.3.25.3 Related Switches . . . 339
15.3.26 –addstatefield . . . 339
15.3.26.1 Syntax . . . 339
15.3.26.2 Description . . . 339
15.3.26.3 Related Switches . . . 340
15.3.27 –encryptcustomsetting . . . 340
15.4 Licensing . . . 340
15.4.1 –getactivationcode . . . 340
15.4.1.1 Syntax . . . 340
15.4.1.2 Description . . . 341
15.4.2 –serialcode . . . 341
15.4.2.1 Syntax . . . 341
15.4.2.2 Description . . . 341

15.4.2.3 Example . . . 341
15.4.3 –showeula . . . 341
15.4.3.1 Syntax . . . 341
15.4.3.2 Description . . . 341
15.4.4 –iaccepteula . . . 341
15.4.4.1 Syntax . . . 341
15.4.4.2 Description . . . 342
15.4.5 –activate . . . 342
15.4.5.1 Syntax . . . 342
15.4.5.2 Description . . . 342
15.4.6 –deactivate . . . 342
15.4.6.1 Syntax . . . 342
15.4.6.2 Description . . . 342
15.5 Tools . . . 342
15.5.1 –simpriocalc . . . 343
15.5.1.1 Syntax . . . 343
15.5.1.2 Description . . . 343
15.5.2 –getembeddedimage . . . 343
15.5.2.1 Syntax . . . 343
15.5.2.2 Description . . . 343
15.5.2.3 Examples . . . 344
15.5.3 –vscode . . . 344
15.5.3.1 Syntax . . . 344
15.5.3.2 Description . . . 344
15.5.3.3 Example . . . 344
15.5.4 –ulsviewer . . . 344
15.5.4.1 Syntax . . . 344
15.5.4.2 Description . . . 344
15.5.5 –cleanuperrorlist . . . 344
15.5.5.1 Syntax . . . 344
15.5.5.2 Description . . . 345
15.6 Advanced . . . 345
15.7 Other . . . 345
15.7.1 –help . . . 345
15.7.1.1 Syntax . . . 345
15.7.1.2 Description . . . 345
15.7.2 –testmain . . . 345
15.7.2.1 Syntax . . . 345
15.7.2.2 Description . . . 346
15.8 Old . . . 346
15.8.1 Work Modes . . . 346
15.8.1.1 `--run` . . . 346
15.8.1.2 `--workflowroot` . . . 347
15.8.1.3 `--vscode` . . . 347
15.8.1.4 `--onlywf` . . . 347
15.8.1.5 `--exec` . . . 347
15.8.1.6 `--teststate` . . . 347
15.8.1.7 `--notbe` . . . 347
15.8.2 Workflows . . . 347
15.8.2.1 Basics . . . 347
15.8.2.1.1 `--new [--wftype]` . . . 347
15.8.2.2 HistoryList . . . 348
15.8.2.3 DataList . . . 348
15.8.2.4 ConfigList . . . 348
15.8.2.5 StateList . . . 348
15.8.2.6 Credentials User . . . 349
15.8.2.7 Credentials App . . . 349
15.8.2.8 SharePoint Permissionsets . . . 349

15.8.2.8.1 --createpermissionsets . . . 351
15.8.3 Tests . . . 351
15.8.3.1 --sendmail="<mail-address>" . . . 351
15.8.3.2 --testmain . . . 351
15.8.3.3 --rcfg . . . 351

16 Global Configruation **353**
16.1 Structure . . . 353
16.2 Complete file kenaflow.conf . . . 354
16.3 Section "AfterSetup . . . 355
16.3.1 adminMails . . . 355
16.3.2 WorkflowRoot . . . 355
16.3.3 encryptionPassphrase . . . 355
16.3.4 activeDirectoryDomain . . . 355
16.3.5 instanceName . . . 355
16.3.6 Full Section Default . . . 356
16.3.7 Related Cmdlets . . . 356
16.4 Section "Licensing . . . 356
16.4.1 licensingMailAddress . . . 356
16.4.2 Serial . . . 356
16.4.3 ActivationCode . . . 356
16.4.4 lastActivationCode . . . 356
16.4.5 lastOffineDeactivationCode . . . 357
16.4.6 Full Section Default . . . 357
16.5 Section "Basic . . . 357
16.5.1 database . . . 357
16.5.2 globalFloodProtectionFile . . . 357
16.5.3 globalFloodProtectionTBE . . . 358
16.5.4 globalWorkflowExecutionTBE . . . 358
16.5.5 globalWorkflowStateTBE . . . 358
16.5.6 ConfigEditor . . . 358
16.5.7 EncryptConfig . . . 358
16.5.8 Branch . . . 359
16.5.9 InfoServerPort . . . 359
16.5.10 ProgramDataFolder . . . 359
16.5.11 CustomLibrary . . . 359
16.5.12 VSCodeDir . . . 360
16.5.13 encryptSerializedWFCONFIG . . . 360
16.5.14 allowPowerShellPlaceholders . . . 360
16.5.15 PluginsEnabled . . . 360
16.5.16 PluginsFolder . . . 360
16.5.17 httpSecurityProtocolType . . . 361
16.5.18 Full Section Default . . . 361
16.6 Section "Storage . . . 362
16.6.1 storageEnabled . . . 362
16.6.2 storagePath . . . 362
16.6.3 storageMaxSize . . . 362
16.6.4 Full Section Default . . . 362
16.7 Section "Monitoring . . . 363
16.7.1 usePerformanceCounter . . . 363
16.7.2 WorkReportHookAddress . . . 363
16.7.3 ClientStartReportHookAddress . . . 363
16.7.4 PingICMP . . . 363
16.7.5 PingWebHook . . . 364
16.7.6 StopCountFailedPings . . . 364
16.7.7 ErrorWebHook . . . 364
16.7.8 FreeDiskSpaceDrives . . . 365
16.7.9 FreeDiskSpaceThreshold . . . 365

16.7.10 FreeDiskSpaceWebHook . . . 366
16.7.11 Full Section Default . . . 366
16.8 Section "Runtime . . . 366
16.8.1 Settings for all workflow types . . . 366
16.8.1.1 MaxExecutionTime . . . 366
16.8.1.2 defaultScriptRunTimeout . . . 367
16.8.1.3 MaxRuntimeOfChildProcess . . . 367
16.8.1.4 VariationOfMaxRuntimeOfChildProcess . . . 367
16.8.1.5 poshQueueSize . . . 368
16.8.1.6 culture . . . 368
16.8.2 Settings related to SharePoint workflows (site workflow, list workflow, state machine worklfow) . . . 368
16.8.2.1 ItemReCheckTime . . . 368
16.8.2.2 ItemBatchCount . . . 369
16.8.2.3 UserProfileCacheLifetime . . . 369
16.8.2.4 Full Section Default . . . 369
16.9 Section "Logging . . . 370
16.9.1 useSerilog . . . 370
16.9.2 serilogBaseTemplateConsole . . . 370
16.9.3 serilogBaseTemplateFile . . . 370
16.9.4 serilogTemplate . . . 370
16.9.5 minConsoleLogLevel . . . 370
16.9.6 minFileLogLevel . . . 371
16.9.7 dontLogIfMainWindowIsHidden . . . 371
16.9.8 logDir . . . 371
16.9.9 logEnabled . . . 371
16.9.10 logOnlyMainProcess . . . 371
16.9.11 logMemMaxAge . . . 372
16.9.12 logFileMaxAge . . . 372
16.9.13 logCleanupRunner . . . 372
16.9.14 logCleanupRunner . . . 372
16.9.15 logFCE . . . 372
16.9.16 logOutput . . . 372
16.9.17 sendOutputToMainProcess . . . 373
16.9.18 logfileLockTimeout . . . 373
16.9.19 logCleanupThreadLogBlocks . . . 373
16.9.20 heartbeatMessageInLog . . . 373
16.9.21 garbageCollectionMessageInLog . . . 373
16.9.22 disableStacktrace . . . 374
16.9.23 useMailHashing . . . 374
16.9.24 mailhashDir . . . 374
16.9.25 mailhashIntervals . . . 374
16.9.26 Full Section Default . . . 374
16.10 Section "Main Mode . . . 375
16.10.1 parallelWFExecutors . . . 375
16.10.2 rerHostEnabled . . . 375
16.10.3 rerHostOnlyMode . . . 375
16.10.4 slots . . . 376
16.10.5 disablePriorityAlgorithm . . . 376
16.10.6 FirstRunTBEAddPercentage . . . 376
16.10.7 Full Section Default . . . 376
16.11 Section "SharePointConnection . . . 377
16.11.1 connectionStartThrottling . . . 377
16.11.2 throttlingRetryCount . . . 377
16.11.3 throttlingInitialDelay . . . 377
16.11.4 throttlingDelayFactor . . . 377
16.11.5 Full Section Default . . . 377
16.12 Section "MultiProcessing . . . 378

16.12.1 testClientRunners . . . 378
16.12.2 retryCount . . . 378
16.12.3 waitBeforeRetry . . . 378
16.12.4 QueueIdleWait . . . 378
16.12.5 parentProcessCheckInterval . . . 378
16.12.6 heartbeatInterval . . . 378
16.12.7 heartbeatTimeout . . . 379
16.12.8 Full Section Default . . . 379
16.13 Section "Lock . . . 379
16.13.1 lockDir . . . 379
16.13.2 skippedItemCacheDir . . . 379
16.13.3 maxWaitForLock_Item . . . 379
16.13.4 maxWaitForLock_List . . . 380
16.13.5 maxWaitForLock_Workflow . . . 380
16.13.6 Full Section Default . . . 380
16.14 Section "Mail . . . 380
16.14.1 useSharePointForSend . . . 380
16.14.2 smptserver . . . 381
16.14.3 smptserverport . . . 381
16.14.4 smptusername . . . 381
16.14.5 smptpassword . . . 381
16.14.6 smptUseSsl . . . 381
16.14.7 smptUseDefaultCredentials . . . 381
16.14.8 mailSender . . . 381
16.14.9 mailReplyto . . . 382
16.14.10 subjectPrefix . . . 382
16.14.11 subjectSuffix . . . 382
16.14.12 Full Section Default . . . 382
16.15 Section "MailDev . . . 382
16.15.1 writeMailsToDiskDuringDebugDefaultSetting . . . 382
16.15.2 alwaysWriteMailsToDiskInsteadSending . . . 383
16.15.3 mailsOnDisk . . . 383
16.15.4 Full Section Default . . . 383
16.16 Section "RER . . . 383
16.16.1 rerRunnerSP2013 . . . 383
16.16.2 rerRunnerSP2016 . . . 384
16.16.3 rerRunnerSP2019 . . . 384
16.16.4 rerRunnerSPO . . . 384
16.16.5 rerRunnerPOSH . . . 384
16.16.6 RemoteEventHandlerName . . . 384
16.16.7 rerQueueDir . . . 385
16.16.8 rerHost . . . 385
16.16.9 rerHostIp . . . 385
16.16.10 rerHostUsesSSL . . . 385
16.16.11 rerPort . . . 385
16.16.12 excludeCurrentUser . . . 386
16.16.13 excludeUsers . . . 386
16.16.14 rerCheckDuringWorkflowExec . . . 386
16.16.15 postponeEventOnScriptError . . . 387
16.16.16 encryptRemoteEvents . . . 387
16.16.17 sslCertThumbprint . . . 387
16.16.18 rerReponseCheckEnabled . . . 387
16.16.19 rerReponseCheckInterval . . . 387
16.16.20 rerReponseCheckTimeout . . . 388
16.16.21 rerReponseCheckFailCount . . . 388
16.16.22 rerReponseCheckEnabledSPRER . . . 388
16.16.23 rerReponseCheckEnabledLINK . . . 388
16.16.24 rerReponseCheckEnabledCDA . . . 388

16.16.25 rerReponseCheckEnabledUDA . . . 388
16.16.26 Full Section Default . . . 388
16.17 Section “cmdletConfic . . . 389
16.17.1 Settings for Cmdlet Invoke-KFFileTemplate . . . 389
16.17.1.1 FileTemplateStartMarker . . . 389
16.17.1.2 FileTemplateEndMarker . . . 390
16.17.2 Full Section Default . . . 390
16.18 Section “Links . . . 390
16.18.1 linkClickAnswer . . . 390
16.18.2 linkClickExpired . . . 390
16.18.3 Full Section Default . . . 391
16.19 Section “DoNotTouch . . . 393
16.19.1 instanceId . . . 393
16.19.2 configFileVersion . . . 393
16.19.3 productVersion . . . 393
16.19.4 val1 . . . 393
16.19.5 val2 . . . 394
16.19.6 Full Section Default . . . 394
16.20 kenaflow.conf . . . 394

17 Workflow Configuration **403**
17.1 Most Important Settings to Start With . . . 405
17.2 General Workflow Settings . . . 405
17.3 Site Workflow Configuration . . . 411
17.4 List Workflow Configuration . . . 413
17.5 State Machine workflow Configuration . . . 418
17.5.1 Workflow State Configuration . . . 422
17.6 Email Workflow Configuration . . . 425
17.7 PowerShell Workflow Configuration . . . 427
17.8 Complete Template . . . 428
17.8.1 Complete file _wfconfig.ps1 . . . 428
17.8.2 Site Workflow Complete Template . . . 428
17.8.3 List Workflow Complete Template . . . 433
17.8.4 State Machine Workflow Complete Template . . . 438
17.8.5 Email Workflow Complete Template . . . 445
17.8.6 PowerShell Workflow Complete Template . . . 450

18 How to **455**
18.1 App ID & App Secret . . . 455
18.1.1 Delete AppCredentials . . . 457
18.2 Using GIT in Workflow Development . . . 457
18.2.1 Introduction . . . 457
18.2.2 Showcase . . . 458
18.2.3 Creating a Git repository . . . 458
18.2.4 Creating the different development stages . . . 459
18.3 Workflow Error When Using Invoke-WebRequest . . . 459
18.4 Get/Set SharEPoint List Item Moderation State . . . 460

19 Open Source Licenses **461**

20 Privacy Policy **465**

21 Imprint **469**
21.1 Contact Information . . . 469
21.2 Responsible for web site content . . . 469
21.3 Disclaimer . . . 469
21.3.1 Liability for Content . . . 469
21.3.2 Liability for Links . . . 469
21.3.3 Copyrights . . . 470

22 Glossary **471**
22.1 *kenaflow* Instance . . . 471
22.2 Program Mode . . . 471
22.3 Execution Lane . . . 471
22.4 Execution Slot . . . 471
22.5 Workflow Priority . . . 472
22.6 TBE = Time Between Execution . . . 472
22.7 PowerShell ISE . . . 472
22.8 Workflow Folder (Workflow Directory) . . . 472
22.9 Workflow Root Folder (Workflow Root Directory) . . . 472
22.10 Workflow Configuration File . . . 472
22.11 Global Configuration File . . . 472
22.12 Data Folder (Data Directory) . . . 473
22.13 Queue Folder (Queue Directory) . . . 473
22.14 Lock Folder (Lock Directory) . . . 473
22.15 Mails Folder (Mails Directory) . . . 473
22.16 Log Folder (Log Directory) . . . 473
22.17 Skip Folder (Skip Directory) . . . 473
22.18 Remote Event . . . 474
22.19 Cmdlet . . . 474
22.20 PowerShell . . . 474
22.21 netsh.exe . . . 474
22.22 kenaflow.exe . . . 474
22.23 kenaflow.runtime.dll . . . 474

23 Final Notes **475**
23.1 About this document . . . 475
23.2 Disclaimer . . . 475
23.3 Brands . . . 475

24 kenaflow Change Log **477**
24.1 [4.0.23] - 2021-11-07 . . . 477
24.1.1 Added . . . 477
24.2 [4.0.22] - 2021-11-04 . . . 477
24.2.1 Added . . . 477
24.3 [4.0.21] - 2021-10-21 . . . 477
24.3.1 Changed . . . 477
24.4 [4.0.20] - 2021-10-08 . . . 478
24.4.1 Fixed . . . 478
24.5 [4.0.19] - 2021-06-09 . . . 478
24.5.1 Fixed . . . 478
24.6 [4.0.18] - 2021-05-24 . . . 478
24.6.1 Fixed . . . 478
24.7 [4.0.17] - 2021-04-12 . . . 478
24.7.1 Fixed . . . 478
24.8 [4.0.16] - 2021-03-05 . . . 478
24.8.1 Added . . . 478
24.8.2 Changed . . . 478
24.9 [4.0.15] - 2021-03-03 . . . 479
24.9.1 Fixed . . . 479
24.10 [4.0.14] - 2021-02-27 . . . 479
24.10.1 Fixed . . . 479
24.11 [4.0.13] - 2021-02-01 . . . 479
24.11.1 Fixed . . . 479
24.12 [4.0.12] - 2021-01-11 . . . 479
24.12.1 Added . . . 479
24.13 [4.0.11] - 2020-12-24 . . . 479
24.13.1 Added . . . 479

24.14 [4.0.10] - 2020-12-18 . . . 479
24.14.1 Added . . . 479
24.15 [4.0.9] - 2020-11-10 . . . 480
24.15.1 Added . . . 480
24.16 [4.0.8] - 2020-11-09 . . . 480
24.16.1 Fixed . . . 480
24.16.2 Changed . . . 480
24.17 [4.0.7] - 2020-09-28 . . . 480
24.17.1 Changed . . . 480
24.17.2 Added . . . 480
24.18 [4.0.6] - 2020-09-14 . . . 480
24.18.1 Fixed . . . 480
24.18.2 Added . . . 480
24.19 [4.0.5] - 2020-07-29 . . . 481
24.19.1 Changed . . . 481
24.19.2 Fixed . . . 481
24.20 [4.0.4] - 2020-01-31 . . . 481
24.20.1 Changed . . . 481
24.21 [4.0.0] - 2020-01-31 . . . 481
24.21.1 Changed . . . 481
24.22 Added . . . 481
24.23 [3.0.17] - 2019-12-12 . . . 481
24.23.1 Added . . . 481
24.24 [3.0.16] - 2019-12-11 . . . 482
24.24.1 Fixed . . . 482
24.25 [3.0.15] - 2019-11-26 . . . 482
24.25.1 Fixed . . . 482
24.26 [3.0.14] - 2019-09-17 . . . 482
24.26.1 Changed . . . 482
24.27 [3.0.13] - 2019-09-17 . . . 482
24.27.1 Added . . . 482
24.27.2 Changed . . . 483
24.28 [3.0.12] - 2019-09-08 . . . 483
24.28.1 Changed . . . 483
24.29 [3.0.11] - 2019-08-20 . . . 483
24.29.1 Fixed . . . 483
24.30 [3.0.10] - 2019-07-29 . . . 483
24.30.1 Fixed . . . 483
24.31 [3.0.9] - 2019-07-26 . . . 483
24.31.1 Fixed . . . 483
24.31.2 Changed . . . 483
24.32 [3.0.8] - 2019-07-25 . . . 484
24.32.1 Fixed . . . 484
24.32.2 Changed . . . 484
24.33 [3.0.7] - 2019-07-24 . . . 484
24.33.1 Fixed . . . 484
24.33.2 Changed . . . 484
24.33.3 Added . . . 484
24.34 [3.0.6] - 2019-07-18 . . . 484
24.34.1 Changed . . . 484
24.35 [3.0.5] - 2019-07-17 . . . 485
24.35.1 Changed . . . 485
24.35.2 Added . . . 485
24.36 [3.0.4] - 2019-07-08 . . . 485
24.36.1 Added . . . 485
24.36.2 Changed . . . 485
24.36.3 Fixed . . . 485
24.37 [3.0.3] - 2019-06-26 . . . 485

24.37.1 Added . . . 485
24.38 [3.0.2] - 2019-06-25 . . . 486
24.38.1 Changed . . . 486
24.38.2 Added . . . 486
24.38.3 Fixed . . . 486
24.39 [3.0.1] - 2019-06-24 . . . 486
24.39.1 Changed . . . 486
24.40 [3.0.0] - 2019-06-03 . . . 486
24.40.1 Added . . . 486
24.40.2 Changed . . . 486
24.40.3 Fixed . . . 487
24.40.4 [3.0.0] - 2018-05-01 . . . 487

kenaflow is a workflow engine for SharePoint, Email and PowerShell.

The PDF can be downloaded here for the most resent version:

https://kenaflow.blob.core.windows.net/release-4-0/kenaflow-manual-latest.pdf

Or a version specific PDF:

https://kenaflow.blob.core.windows.net/release-4-0/kenaflow-manual-latest.pdf

! The permanent document location for *kenaflow* **4.0** is: ! https://doc.kenaflow.com/v4-0

! The permanent document location for *kenaflow* **3.0** is: ! https://doc.kenaflow.com/v3-0

All workflows are written in PowerShell. Also the configuration of each workflow is a PowerShell script.

Please read the article *The Basics* for details.

We have also prepared an article about the background of workflows and their meaning for the ongoing "digitalization" process: *Workflows and Digitalization*

Some shortcuts to sections of this documentation:

- *Cmdlet Reference*
- *Global Configuration Reference*
- *Workflow Configuration Reference*
- *Commandline Reference*

The following articles will give you an overview about *kenaflow*:

- *Workflows and Digitalization*: Our point of view regarding to the topic.
- *Editions*: What editions of *kenaflow* are available?
- *The Basics*: The very basic information about *kenaflow*.
- *Architecture*: The architecture of *kenaflow* regarding to scalability.
- *The Details*: The architecture and runtime behaviour of a *kenaflow* instance.

! This documentation is written and maintained by the creator of kenaflow, Ingo Karstein, with support from Markus Hottenrott of uniforx. ! ! Ingo is a SharePoint Consultant and Developer based in Leipzig, Germany. He is in the business since 2000, self employed in 2011, became *Microsoft Certified Master: SharePoint 2010* in 2012 and *Microsoft Certified Solutions Master: SharePoint (Charter)* in 2013, and founded kenaro in October 2015.

CHAPTER

ONE

WORKFLOWS AND DIGITALIZATION

Since several years our enterprise and its coworkers are consulting in the field of Microsoft SharePoint and related technologies. Our tasks are the conception, introduction, installation, support and extension of all SharePoint related issues.

We are often involved in the creation of digitized workflows for our customers. The data and information are stored in SharePoint. The workflows should be implemented on this basis. In most cases, it is necessary to include different people in the process at different times, for example, “decision makers”, “deputies”, “information procurers”, “purchasers”, and so on. This is because, the data usually remains in a (virtual) location, but they “move” due to their assignment to specific people through the company.

Workflows are the work processes, in which information is processed by man and machine

In order to realize workflows with SharePoint, additional solutions such as *kenaflow* are not necessarily needed. However, the built-in options are usually not convenient or adequate. Therefore, it makes sense to rely on specialized tools as soon as digitalization is to be advanced across the board.

There are some workflow software publishers for SharePoint, but they all take the same approach of putting workflows in the hands of “Power Users”. “Power Users” are employees of the company who are not necessarily members of the IT department. These workflow software publishers suggest, that implementing and modelling workflows is so easy that every employee can do it. To create workflows there are designed tools: Applications with a graphical interface with which workflows can be “clicked together”.

At first glance, that may seem like a good thing. You drag a “box” from the menu onto the desktop and connect it to other “boxes”. But then comes the moment when the tide turns: the “box” must be clicked to configure the workflow action. In face of what the “Power User” sees at this point, they are usually overwhelmed: fields with setting options, selection lists, fields for URLs, fields for IDs or even configuration texts in XML and so on. At this point, it becomes clear that the complexity of a workflow cannot be brought into a simple, clicked structure by means of a magic tool.

Developing workflows is more the job of professionals/programmers than the task of employees from purchasing, personnel management or controlling.

The mentioned tools and software are expensive, because the development of graphical applications is complex and complicated, which is why they can only be configured up to a certain degree.

However, we have chosen a different approach which, in our view, is better suited for reality and better supports those who, in reality, are capable of implementing the workflows from the technical aspect:

We developed *kenaflow*.

A workflow platform aimed at administrators and developers.

The basis for this is PowerShell, a scripting language that is explicitly aimed at administrators who should be able to manage the server landscapes automated.

PowerShell is an established factor in the SharePoint context because SharePoint is configured via PowerShell. A professional SharePoint consultant will do almost all of the configuration through PowerShell and just open the “Central Administration” to check the progress.

PowerShell is also becoming more important in other areas of everyday IT, and without PowerShell, SharePoint can hardly be adapted to the real needs of a business.

So we thought it would be practical and logical to develop SharePoint workflows based on PowerShell.

What is missing, however, is the software component that handles the execution of the PowerShell scripts: **executing the right script in the right context at the right time.**

Translating a single workflow into a single script is not difficult. However, implementing many workflows in many scripts and managing them is difficult. So a way has to be found to deal with this problem. Furthermore, it must also be easy to maintain.

kenaflow defines a "Workflow Folder", each containing all the files necessary for a workflow. These workflow folders are located in the workflow root.

One of the strengths of *kenaflow* is the ability to version workflows: any workflow folder or the workflow root can be easily linked to GIT or any other source code management-, synchronization software or backup solution, and it can be retracted at any time how workflows are changing.

Finally, when it comes to running several or even a lot of workflows, the topic of "parallelization" is important.

kenaflow offers the possibility to run many workflows simultaneously. The degree of parallelization can be controlled by the *kenaflow* administrator.

In terms of scaling, *kenaflow* also offers the possibility of running the software on several computers at the same time. The *kenaflow* instances synchronize workflow processing among each other.

Another advantage is that *kenaflow* can be executed on a single computer in multiple instances, e.g. to realize workflow development and workflow productive operations on the same server. The *kenaflow* instances do not interfere with each other.

Since workflow development comes close to the programming of software, it is natural that known procedures from software development have been implemented in *kenaflow*.

Versionability has already been mentioned. Another aspect is the "debugging", i.e. the examination of program code during execution, to check the execution itself or to control the contents of variables. - Debugging is an essential tool. *kenaflow* implements two options and both allow you to go into the PowerShell scripts. For this purpose "PowerShell ISE" and/or "Visual Studio Code" are/is used. The PowerShell ISE has the advantage of being part of the Windows operating system and is available everywhere. - Details can be found in the chapter "Debugging".

CHAPTER

TWO

EDITIONS

2.1 *kenaflow* Standard

This edition is not limited at all. It can be installed on a single server multiple times.

Unlimited workflows (limited only by your hardware) on the server.

Unlimited *kenaflow* instances on the server.

Unlimited remote runner processes (limited only by your hardware).

Plugins are available.

1 license is required per server.

2.2 *kenaflow* Developer

This edition of *kenaflow* is intended to develop workflows "locally" on the developers machine. You should not it for production.

This edition is not limited in the workflow feature set.

But you can only have 1 *kenaflow* instance per server with this license.

You can have a maximum of 5 workflows at all.

You can have a maximum of 5 "execution lanes".

You can have a maximum of 1 "rer runner mode" per supported platform.

Plugins are available.

The licence is valid for 1 year from time of subscribing.

1 license is required per development machine.

In DEVELOPER edition the global configuration file is not encrypted.

In DEVELOPER edition remote events are not stored encrypted on disk.

In DEVELOPER edition "rich links" generated with `Get-KFLink` are not encrypted!

2.3 *kenaflow* Free

This edition of *kenaflow* is intended to get familar with the engine. You can test it. But you should not use it for development.

This edition is not limited in the workflow feature set.

But you can only have 1 *kenaflow* instance per server with this license.

You can have a maximum of 4 workflows at all.

You can have a maximum of 1 "execution lane".

Plugins are not available!

The licence is valid for 3 months from time of subscribing.

In FREE edition the global configuration file is not encrypted.

In FREE edition remote events are not stored encrypted on disk.

In FREE edition "rich links" generated with `Get-KFLink` are not encrypted!

CHAPTER

THREE

THE BASICS

Our approach is simple. kenaflow_ is designed for administrators and IT professionals.

Workflows are developed in PowerShell. The workflow developer uses his PowerShell skills and know-how.

kenaflow provides the runtime environment for the workflow scripts.

The workflow scripts are developed in PowerShell ISE or with Visual Studio code. The "ISE" is part of the Windows operating system.

We have integrated Visual Studio Code into the *kenaflow* program folder. You can start the copy of Visual Studio code intended for a *kenaflow* instance with `kenaflow.exe --vscode`.

kenaflow allows you to debug workflow scripts.

Each workflow is controlled by its own configuration file. The workflow configuration file defines WHAT the workflow does, WHERE the workflow works, and WHAT data (list items or mail) the workflow will work with.

At the right time, *kenaflow* PowerShell scripts run for the right data.

kenaflow controls the loading process of workflows.

kenaflow starts workflows in parallel.

kenaflow is able to receive "remote events" from SharePoint or third-party systems.

kenaflow supports SharePoint 2013, SharePoint 2016, SharePoint 2019 and SharePoint Online.

kenaflow supports Exchange 2007, 2010, 2013 and Online as well as IMAP servers and POP3 mailboxes.

Workflows built with *kenaflow* are versionable.

Please read the article *Architecture* next.

CHAPTER

FOUR

ARCHITECTURE

In this section, we would like to show you how *kenaflow* is to be scaled.

The basic principles during development of *kenaflow* were:

- Focus on major customers ("Enterprise")
- Complete automation of the application *kenaflow*
- Encapsulation of the complexity of the workflow execution in the engine for the simplest possible handling of the topic by customers
- Alignment of the software to the skills and needs of administrators who 1) support and 2) create workflows with *kenaflow*.

If you finished reading please continue with article *"The Details"*.

4.1 Enterprise

kenaflow offers two types of scaling:

1. Multiple instances of *kenaflow* **can run simultaneously** on a server. (Vertical scaling)
2. Multiple servers can share the execution of workflows. (Horizontal scaling)

!! In FREE and DEVELOPER edition of *kenaflow* you can only run 1 instance of *kenaflow* on 1 machine.

4.1.1 Vertical Scaling = Instances

kenaflow offers a simple setup and a manageable number of storage locations where data is stored:

- Program folder
- Program Data folder
- Workflow Root folder

By default, this is the case for a single instance:

- Program folder: `C:\Program Files\kenaflow`
- Program Data folder: e.g. `C:\ProgramData\kenaflow\instance_8a040c56-4343-4f7a-9713-9aa56b1dc` whereby the GUID ("8a040c56-4343-4f7a-9713-9aa56b1dcc21") is once assigned during installation.
- Workflow Root folder: e.g. `C:\Workflows`

With multiple instances, it could look like this:

- Instance 1
  - Program folder: `C:\Program Files\kenaflow-dev-v1.0`
  - Data folder: `C:\ProgramData\kenaflow\instance_62a289df-fded-4fe9-9bc4-4cd7b01561c3`

  - Workflow Root folder: `C:\Workflows-dev`
- Instance 2
  - Program folder: `C:\Program Files\kenaflow-prod-v1.0`
  - Data folder: `C:\ProgramData\kenaflow\instance_74dcd84e-8bb3-44ba-b2b4-402d2e63b667`
  - Workflow Root folder: `C:\Workflows-prod`
- Instance 3
  - Program folder: `C:\Program Files\kenaflow-next-v1.1`
  - Data folder: `C:\ProgramData\kenaflow\instance_279afb48-0e55-42bc-89dc-d03815e3c950`
  - Workflow Root folder: `C:\Workflows-next`

Only 1 *kenaflow* license is required for this implementation.

Here, three instances of *kenaflow* run on 1 server.

- "Instance 1" is used for developing workflows.
- "Instance 2" is used for executing productive workflows.
- "Instance 3" is used to test the next version of *kenaflow*.

Any version of *kenaflow* can exist side by side on a server.

A separate execution account can be used for each *kenaflow* instance.

4.1.2 Horizontal Scaling

kenaflow can be installed on multiple servers.

A separate *kenaflow* instance is required for each server.

The program folder is e.g.

- Server 1: `C:\Program Files\kenaflow`
- Server 2: `C:\Program Files\kenaflow`
- Server 3: `C:\Program Files\kenaflow`

However, the Data folder and Workflow Root folder are placed on a network share to which the *kenaflow* instances of all three servers have access. This is ensured via the used service account.

The three servers in this example use the same Workflow Root folder. Each *kenaflow* server locks a workflow that is currently being executed using a lock file in the data folder. This means that the other servers will not execute this workflow in parallel and will, therefore, execute other workflows.

4.1.3 Vertical and Horizontal Scaling Together

Both scaling methods can be used in parallel.

CHAPTER

FIVE

THE DETAILS

Here we descibe who a *kenaflow* instance works on the *kenaflow* server.

If there are multiple *kenaflow* instances on the server they all behave on their own in the same was as described here.

!! In FREE and DEVELOPER edition of *kenaflow* you can only run 1 instance of *kenaflow* on 1 machine.

As described in article *Architecture kenaflow* has three important folders on the file system:

- Program folder
- Program Data folder
- Workflow Root folder

In the "program folder" is a file named "kenaflow.conf". That contains the so called global configuration. The file is encrypted with a server specific key that is provided by Microsoft Windows. The file cannot be transfered to other machines. - To decrypt it you can use program switch `--decryptconfig`.

The global configuration can be edited with this command: `kenaflow.exe --editconfig`. (Details: `--editconfig`).

To execute *kenaflow* for automatic workflow execution the program is started with this program switch: `kenaflow.exe --run`. This executes the so called "main mode" of *kenaflow*.

If it is started e.g. by the Windows Scheduler (*Automatic Execution*) the additional switch `--norestart` is used. It prevents *kenaflow* the "main mode" to restart automatically, e.g. in case of changes to the global configuration.

The "main mode" is runs in its own process in Windows. It controls the execution of workflows, starts sub processes and monitors them.

During automatic workflow execution different process types ("modes" of `kenaflow.exe`) are required:

1. The "main mode" as described. (`kenaflow.exe --run`)
2. The "rer global mode" (`kenaflow.exe --rer`): This registers several http endpoints to receive remote events. There is only one such process for a *kenaflow* instance.
3. The "rer runner mode" (e.g. `kenaflow.exe --rerrunnersp2019`): For every platformat that supports remote events with kenaflow (all types of SharePoint workflows and PowerShell workflows) there by default at least 1 such "rer runner mode" process. It monitors the remote event queue (see below) and processes incoming remote events that are specific to the platform of the "rer runner mode". E.g. the "rer runner mode" process started with `kenaflow.exe --rerrunnersp2019` does only process remote events for workflows that have specified `SP2019` as their platform in `_wfconfig.ps1`.)

   It is possible disable a certain "rer runner mode" process for a platform. E.g. if you do not connect to SharePoint Online you can disable the "SharePoint Online rer runner mode". This is done in global configuration.It is also possible to configure more than 1 "rer runner mode" per platform. E.g. if you have much remote events coming from "SharePoint 2019" you can configure 2 or more "rer runner mode" processes to execute in parallel. Each of them will process remote events from the queue. But every remote event is processed by exactly on "rer runner mode" process at a time. If a remote event gets postponed it's maybe executed by another "rer runner mode" process next time.

4. The "exectution mode" (e.g. `kenaflow.exe --exec --workflow "<workflow-forlder>"`): For every workflow execution a separate, new and fresh process is started. For security reasons. If such a process fails it gets terminated by the "main mode" process

    There can be multiple "execution mode" processes at one time. How much is decided by the main mode. But the maximum of parallel running "execution mode" processes is configured in global configuration. Default is 5. We call this "execution lane".

!! In FREE and DEVELOPER you can have only one "rer runner mode" per platform!

The main mode process starts the sub processes. It monitors that they are running.

The sub processes of all types monitor the main mode process. If it gets terminated they terminate itself ASAP.

The sub processes communicate with the main mode process using "inter process communication".

The sub processes send heartbeat messages to the main mode process to tell their "owner" that they are alive. If they do not send such a message they get terminated by the main mode process.

If a subprocess was terminated the main mode starts them again. E.g. if you configured 5 "rer runner modes" for SharePoint Online and one is terminated for some reasons the main mode process will start 1 new "rer runner mode" for SharePoint Online.

All enabled workflows - they have a setting of `Enabled = $true` in their `_wfconfig.ps1` - are queued for execution. This queue is maintained by the main mode process. The first workflow in the queue is executed by a new "execution mode" process started by the main mode. It gets started if:

1. an "execution lane" is available and
2. the workflow configuration "TBE" and its last run timestamp allow the workflow to run. As described before there are a number of parallel "execution lanes". If they are full (e.g. 5 workflows already running in parallel) the next workflow must wait. If a workflow (it's execution mode process) end the next waiting workflow will take the free lane.

Because the engine cannot tell how long the execution of a certain workflow will take it's not possible to exactly schedule a workflow to run.

To control the workflow execution each workflow has a "TBE" setting. This means: "Time Between Execution". Its an amount of seconds that must elapse between two **starts** of the workflow before it runs again. - If TBE is 0 it will be queues to "run again ASAP". If TBE is < 0 it will never run by time.

The workflow queue is operated by a simple "first in first out" mechanism. By no means! The order of the workflows is permanently recalculated, as soon as there is a free "execution lane".

First: Execution lanes can be grouped into "slots". This is an advanced topic, described here: *Execution Slots*. Each slot has its own calculated queue. By default there is just 1 such "slot" with id 0. If not configured otherwise all workflows are executed in this slot.

Second: Each workflow can have a "priority" setting, to be configured in `_wfconfig.ps1`. By default each workflow has a priority of 0 which means that all have the same priority. (Read this: *Workflow Priority*.)

Each time an execution lane (of a "slot") is free for a new workflow from the queue, the waiting workflows are sorted by:

- Their priority
- Their time *in* the queue since their last run.

There is some maths behinde the scene. It takes care that also low priorized workflows get executed some times also if high priorized workflows with a low "TBE" are waiting.

In the Windows Task Manager you can see the running processes. Make sure you enable the column "Command-line" in Task Manager: Switch to tab "Details". Right-Click the headline of the process list, e.g. on "PID". Click "Select Column". In the upcoming dialog check "Command line".

Some more things:

1. If the global configuration "kenaflow.conf" changes all processes recognize this and try to stop ASAP.
2. If the Windows OS is shutting down all *kenaflow* processes try to stop ASAP.

3. If a file named ".update-pending" is created in the program folder this indicates the *kenflow* processes that an update is pending. They all try to stop ASAP.

CHAPTER

SIX

NEW FEATURES IN 3.0

There are some new features in *kenaflow* 4.0 and many improvements.

- *kenaflow Agent* for synchronous and asynchronous remote PowerShell execution.
- some new cmdlets
- better performance

Here are the new features of *kenaflow* 3.0 and many improvements.

6.1 Multiple Webs and Lists

SharePoint workflows - *list*, *state machine* and *site* - can now be connected to multiple SharePoint webs. Therefore a new `Connection` property was introduced to `_wfconfig.ps1`.

For *list* and *state machine* workflows it is also possible to connect each web to a set of SharePoint lists. (This is of cause not available for *site* workflows.)

Example:

```
> Connection = @(
>   @{
>     Web = "https://firstweb.sharepoint.farm";
>     List = @("List 1", "List 2");
>     AlternateWebUrls = @();
>   },
>   @{
>     Web = "https://secondweb.sharepoint.farm";
>     List = "List 3";
>     AlternateWebUrls = @();
>   },
>   @{
>     Web = "https://thirdweb.sharepoint.farm";
>     List = @("List 4", "List 5");
>     AlternateWebUrls = @();
>   }
> )
```

The simple `_wfconfig.ps1` settings `Web` and `List` from *kenaflow* 2.0 are compatible with this new feature. But you cannot use then together with `Connection`.

This, for example, would cause an validation error:

```
> Web = "https://firstweb.sharepoint.farm";
> List = "List 1";
>
> Connection = @(
>   @{
>     Web = "https://firstweb.sharepoint.farm";
>     List = @("List 1", "List 2");
>   }
> )
```

6.2 Item Change Hash Fields

This belongs to SharePoint *list* and *state machine* workflows.

You can now specify "item change hash fields" in `_wfconfig.ps1`. This means, that after processing an item the engine calculates a hash based on the current item values and stores them in *the kenaflow Field*. Before the next processing it calculates the hash of the item and only executes the workflow if something in the specified fields was changed.

In case of *state machine* workflows this setting can be overwritten for each state. The default from the workflow "top level" configuration is used if nothing is specified. Each state can have its own hash value stored in the *kenaflow* field or all use the same hash value. This can be configured as workflow level in `_wfconfig.ps1`.

Please read the article *State Machine workflow Configuration*.

6.3 Returning Data from Workflows using Remote Events

Its now possible to **trigger** a workflow and get data back from the triggered workflow.

Please read this article: *Trigger*.

Its now possible to get data from all workflow types except *email workflows* using remote events of type "unstructured data".

Please read this article: *Remote Event*.

6.4 Get an Answer from a Workflow using "Rich Links"

Its now possible to generate "rich links" to call workflows from the browser - e.g. when clicking a link in an email send by a workflow. You can transfer serializable data to the "linked" workflow. And it is possible to get data back from the workflow script, e.g. to give the use a processing result in the browser. Therefore the "linked" workflow can return HTML data.

Please read this article: *Links*.

6.5 Remote Event Debugging

There is a complete new way to debug - or: “simulate” - remote events.

Please read *Debugging* and *Remote Event*.

CHAPTER

SEVEN

NEW FEATURES IN 4.0

There are some new features in *kenaflow* 4.0 and many improvements.

7.1 Agents

kenaflow Agent for synchronous and asynchronous remote PowerShell execution.

7.2 Performance

Performance improvements.

7.3 Permanent Workflows

Workflows can be configured to run permanently.

7.4 Virtual Workflows

Especially SharePoint workflows can be configured to run on multiple webs and lists. Each workflow will be handeled as a virtual workflow that is queued separately.

7.5 PreProcessing Script and PostProcessing Script

These scrips can be specified in `_wfconfig.ps1`.

If specified they are executed *before* and/or *after* the workflow is executed.

7.6 Pre-Defined Variables

= *kenaflow* 4.0.22

Pre-Defined Variables

7.7 Storage Workflow

7.8 Pre-Configured Execution Environment

7.9 3 “Excel” related Cmdlets

= *kenaflow* 4.0.10

- `Import-KFExcelFile`
- `New-KFExcelFile`
- `Update-KFExcelFile`

CHAPTER

EIGHT

SETUP

To develop workflows for SharePoint 2013, 2016, 2019, SharePoint Online or email accounts such as Exchange, IMAP or POP3 with *kenaflow*, the software must be installed and configured on a computer.

The following sections describe:

1. What the requirements for installing the software are.
2. Which software is **incompatible** with *kenaflow*.
3. How to install *kenaflow*.
4. What changes are made to the system by the *kenaflow* installation.
5. How the license agreements are displayed and accepted.
6. How the software is activated.
7. Which steps are **necessary** before the first start.

8.1 Before You Start

In the following sections, you will learn how to prepare and perform the installation of *kenaflow*.

- *Prerequisites*

  Information about the Windows OS and required software

- *Incompatible Software*

  Information about incompatible software that is not allowed on a *kenaflow* server.

- *System Changes*

  Information about the system changes that are made by *kenaflow*.

8.1.1 Prerequisites

8.1.1.1 Prerequisites

8.1.1.1.1 Hardware

It is recommended that you run *kenaflow* on **a virtual machine** to match the resource requirements to the workload.

The following requirements apply to the physical or virtual hardware:

Minimal System Requirements

- 2 CPU or cores
- a minimum of 8GB RAM

Recommended Configuration For A Medium *kenaflow* Installation

- 4 CPU or cores
- a minimum of 12 GB RAM

8.1.1.1.2 Operating System

kenaflow runs exclusively on computers with a **64-bit Windows** operating system. Please make sure that you meet this basic requirement.

kenaflow normaly runs on server operating systems:

- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019

!! "Core" editions of Windows Server are not testet currently. Mainly this is because workflow development requires a development tool such as PowerShell ISE or Visual Studio Code. !! !! You can try it by installing Server Core App Compatibility Feature on Demand (FOD) (a Microsoft page) on "core" and see if it works for you. - We would be happy about feedback!

kenaflow also runs on client operation systems:

- Windows 7
- Windows 8
- Windows 8.1
- Windows 10

It is not recommended to run *kenaflow* on a computer with other software products. Depending on the workload and various settings in the software, *kenaflow* can be very resource-intensive.

In particular, no web server (e.g., IIS) should be running on the same computer.

8.1.1.1.3 Server Side Storage

The drive on which *kenaflow* is installed must have about 2GB free space for the program files.

kenaflow generates logfiles during execution, which can become very large depending on the load. For this, 30GB of space should be provided.

By default *kenaflow* will be installed in the following directory `C:\Program Files\kenaflow`.

In the default configuration, *kenaflow* stores the execution data - e.g. log files in a subdirectory of `C:\ProgramData\kenaflow`.

8.1.1.1.4 Software

It is recommended to install *kenaflow* on an English-language operating system.

For some of the software components listed below, you can choose from a variety of languages. Although *kenaflow* itself is currently only offered in English, other language variants of the software components listed below may be used.

kenaflow **requires** .NET Framework 4.8.

.NET Framework 4.8:

> https://dotnet.microsoft.com/download/dotnet-framework/thank-you/net48-offline-installer (Download starts after 30 seconds automatically!) Direct link: https://download.visualstudio.microsoft.com/download/pr/014120d7-d689-4305-befd-3cb711108212/0fd66638cde16859462a6243a4629a50/ndp48-x86-x64-allos-enu.exe

PowerShell 5 as a part of the Windows Management Framework. *In Windows 10 and Windows Server 2016 this is already installed.*

For other operating systems, you can download additional packages at the following address. These are provided by Microsoft:

> https://www.microsoft.com/en-us/download/details.aspx?id=54616

Visual C++ Runtime 2015 in the current version:

> https://download.microsoft.com/download/9/3/F/93FCF1E7-E6A4-478B-96E7-D4B285925B00/vc_redist.x64.exe

8.1.2 Incompatible Software

The following software is incompatible with *kenaflow*.

8.1.2.1 SharePoint PnP PowerShell

On a computer running *kenaflow*, **do not use** the PowerShell module for SharePoint from the "Patterns and Practices" framework.

This module can be found here:

https://github.com/SharePoint/PnP-PowerShell

This module is integrated into *kenaflow* and is loaded dynamically at runtime. There are several versions of the module, each for one of the SharePoint versions 2013, 2016, 2019 and Online. - *kenaflow* loads the correct version, but from its own location. An installed version in the operating system will cause *kenaflow* not to work.

However, to take advantage of the capabilities of the PnP PowerShell modules, we have developed a PowerShell cmdlet that you can use to load the *kenaflow* version of the modules.

```
Start-KFSharePointModule -SharePointVersion sp2013
```

8.1.2.2 Internet Information Server (IIS)

The Windows web server "IIS" itself is in principle compatible with *kenaflow* but you need to configure it not to interfer with the *kenaflow* registered http endpoints for *remote events*.

8.1.3 System Changes

The *kenaflow* setup performs some changes on the computer it is running on. Here you find information about these changes.

The *kenaflow* setup creates an unique "instance ID" and adds it to the global configuration file of the *kenaflow* instance. This ID is used for some system modifications.

8.1.3.1 Windows Registry

The *kenaflow* setup creates a key in the Windows registry where the program directory is stored. The registry key can be found here:

```
HKEY_LOCAL_MACHINE\SOFTWARE\kenaflow\<instance-id>
```

With this Windows Registry key you can check wether *kenaflow* is installed - and in which folder it is installed.

8.1.3.2 Program Folder

kenaflow binary files, its configuration and some other files are installed in a folder on disk that we call "program folder".

The default setup path is:

`C:\Program Files\kenaflow`

After setup it is required to execute program mode `--securing` like this:

```
kenaflow.exe --securing --account <service-account>
```

A one task this will set *list*, *read* and *execute* permissions for the `<service-account>` on the program folder.

The program mode `--securing` can be executed multiple times if the instance has multiple service accounts for execution.

8.1.3.2.1 Program Data Folder

_kenaflow needs to store data files during execution to a folder we call "program data folder".

The instance ID is used as part of the sub folder name.

E.g.: `C:\ProgramData\kenaflow\instance_74dcd84e-8bb3-44ba-b2b4-402d2e63b667`

After setup it is required to execute program mode `--securing` like this:

```
kenaflow.exe --securing --account <service-account>
```

A one task this will set *list*, *read*, *create*, *write*, *delete* and *execute* permissions for the `<service-account>` on the program folder.

The program mode `--securing` can be executed multiple times if the instance has multiple service accounts for execution.

8.1.3.3 http(s) Endpoints

kenaflow is able to receive data from "remote" systems such as SharePoint - or other third party applications. (Please read *Remote Event* for details.)

Therefore *kenaflow* offers http(s) endpoints.

These endpoints need to be secured.

After the installer has run you need to execute `--securing` to do this.

The cmdlet opens the endpoint `http://+:<port>/<instance-id>` for "All Authenticated Users".

Therefore the following command is executed by the program mode `--securing`:

```
    netsh http add urlacl url=http://+:<port>/<instance-id> listen=yes user="NT␣
↪Authority\All Authenticated Users"
```

- `<port>` is configured in the global configuration.
- `<instance-id>` is optional. It's usage is configured in the global configuration.

If multiple instances of *kenaflow* are running on the same machine, then must be used or all instances use a different port.

8.2 Installation

kenaflow can be installed multiple times on the same computer. Every installation of *kenaflow* on a computer is called an **instance**.

It is not relevant which program versions of *kenaflow* are installed on parallel machines.

We chose a very simple setup mechanism for *kenaflow* to easily enable the installation of different versions.

Its almost a "copy setup". The only thing that is done only in the installer is to write a key into the Windows Registry. This key contains the binary location of the *kenaflow* instance.

The following instructions must be carried out separately **for each instance** of *kenaflow*. If there are several instances on the same computer, make sure that different installation directories are used.

It must be ensured that all *prerequisites* for the installation of *kenaflow* are met.

8.2.1 Installation

After installing the prerequisite software components, you can proceed with the installation of *kenaflow*.

1. Download the setup as a ZIP file from the following address:

- Standard Edition: https://kenaflow.blob.core.windows.net/release-4-0/kenaflow.latest.zip
- Developer Edition: https://kenaflow.blob.core.windows.net/developer-4-0/kenaflow.latest.zip
- Free Edition: https://kenaflow.blob.core.windows.net/free-4-0/kenaflow.latest.zip

1. Unzip the file into a folder of your choice.
2. Start a command line (`cmd.exe`) as administrator ("Run as Administrator"). Do not use "PowerShell" because the installer will not run if a running process "powershell.exe" or "powershell_ise.exe" is found on the computer.
3. Change to the folder to which you have unzipped the contents of the ZIP file.
4. Start `kenaflow.install.exe` with the `-i` or `--install` parameter and, optionally, the `-d <path>` parameter to specify a destination directory.

The installation is being carried out.

Finally, the setup opens the global configuration file of the new *kenaflow* instance in an editor (typically `notepad.exe`)

The global configuration file only applies to this instance of *kenaflow*. So if you have more instances of *kenaflow* on the same computer, they each have their own global configuration files.

Now proceed with *Basic Configuration*.

8.2.2 Complex Scenarios - Some Notes

In advanced setup scenarios, e.g. when the installation is completely automated, it is possible to use a ready-made global configuration file by specifying it with the program switch `--configfile <file-name>` during installation.

If the specified configuration file contains a "blank" instance ID (`00000000-0000-0000-0000-000000000000`), setup will generate a new one. If the instance ID in the specified configuration file is not "empty", then this ID will be used.

It is possible to prevent the opening of the editor at the end of the installation with the `--noeditconfig` program switch.

8.3 Basic Configuration

After installing a *kenaflow* instance the global configuration has to be customized.

kenaflow offers many configuration options as engine level that influence the execution of the workflow. However, just a few are important for a quick start. Most of them you will never touch, but we think that is better to have the option to configure it in detail.

The basic settings in the global configuration we grouped into the *section "AfterSetup"*.

```
1  <?xml version="1.0" encoding="utf-8"?>
2 <config>
3   <section name="aftersetup">
4     <entry key="adminMails" type="System.String" comment="split by ;"></entry>
5     <entry key="WorkflowFolder" type="System.String" default=""></entry>
6   </section>
```

`adminMails` contains a list of email addresses to which *kenaflow* sends messages when problems occur.

`WorkflowRoot` is the main folder in which the actual workflows are stored. The user account under which *kenaflow* is executed must have read and write access there.

For further instructions see the notes on the program switch `--securing`

`encryptionPassphrase` defines the password for the encryption of the workflow-data. **The password must not be changed after the first operation.**

`activeDirectoryDomain` is defining the "Full Qualified Domain Name" (FQDN) of the domain, on which *kenaflow* is executed.

8.3.1 SMTP Settings

Also of importance are the settings in the *section "Mail"*. In this section, the SMTP server to be used is configured.

!! We have a "typo problem" in the naming of the settings. For some reasons that are hidden to us we used "smpt" instead of "smtp" several times. Currently we cannot change it to not interfer our current customers.

```
49  <section name="mail">
50    <entry key="smptserver" type="System.String"></entry>
51    <entry key="smptserverport" type="System.Int32" default="587"></entry>
52    <entry key="smptusername" type="System.String"></entry>
53    <entry key="smptpassword" type="System.String"></entry>
54    <entry key="smptUseSsl" type="System.Boolean" default="true">true</entry>
55    <entry key="mailSender" type="System.String"></entry>
56    <entry key="mailReplyto" type="System.String"></entry>
57  </section>
```

`smptserver` defines the DNS name or the IP adress of the SMTP server which is being used.

`smtpserverport` is the port that will be used.

`smptusername` names the user, who shall connect/authentificate with the SMTP server.

`smptpassword` specifies the password for the user.

`smptUseDefaultCredentials` specifies whether the "default credentials" of the user executing *kenaflow* should be used to log on to the SMTP server if the **SMTP credentials are unspecified** (`smptusername` and `smptpassword`). If YES (`true` = default), then "Windows Integrated Authentication" is used for NTLM authentication.

`smptUseSsl` specifies whether SSL should be used for communication with the SMTP server.

`mailSender` shows the sender address used in the mails.

`mailReplyTo` tells the e-mail recipient to which e-mail address he can reply.

8.4 Before the First Start

Before the first start of *kenaflow* the user has to

1. Read the license agreement
2. Accept the license agreement
3. Activate the software…
    1. Online or
    2. Offline

8.4.1 Attention

If you want to run a second / third / … instance on a computer on which a *kenaflow* instance is already installed and activated, only the steps listed below are necessary:

1. Enter the *Serial Code*
2. Read and Accpet the *License Agreement*
3. Set up *Service Account*
4. Set up *Automatic Execution*

8.4.2 Serial Code

To use *kenaflow* it must be activated.

The following steps must be carried out *before* activation:

1. The serial code has to be assigned (as descript in the current article)
2. Read and accept the license aggreement (*License Agreement*)

To activate *kenaflow*, you must first connect the serial code on the computer to *kenaflow*.

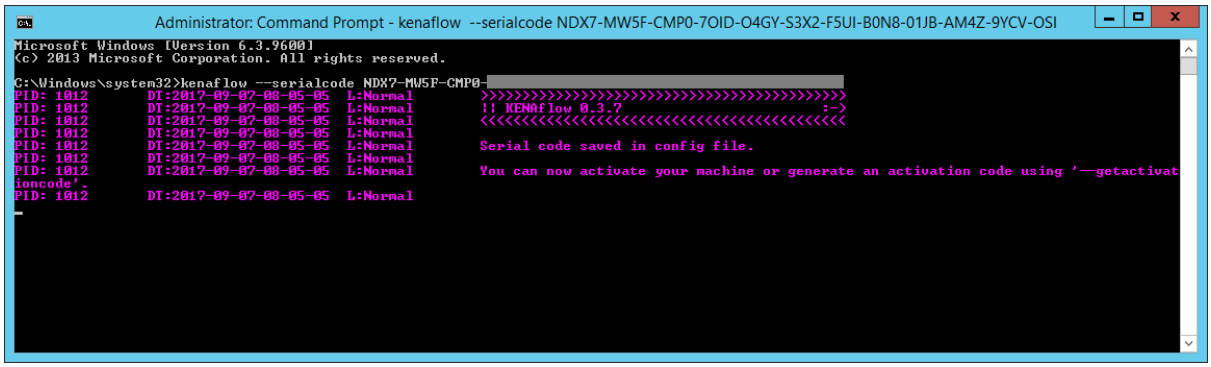
Therefore execute the following command

```
kenaflow.exe --serialcode "<serial-code>"
```

... whereby you replace the `<serial-code>` with the code you received on purchase.

If you run multiple instances of *kenaflow* on one computer, then this command must be executed for each of these instances!

In advanced scenarios, the serial code can be included in the configuration file passed during installation.



Proceed with reading and accepting the *license agreement*

8.4.3 License Agreement

Before *kenaflow* can be used, the license agreement ("EULA") must first be read and then accepted.

This is done using the following program calls:

```
kenaflow.exe --showeula
```

and after

```
kenaflow.exe --iaccepteula
```

These program calls cannot be executed together.

Proceed with *Online activation* or *Offline Activation*.

8.4.4 Online Activation

To use *kenaflow* it has to be activated. To do so you have already assigned a *serial code* and read and accepted the *license agreement*.

kenaflow offers both **automatic online activation** and manual offline activation. The Offline activation is described in *this article*.

Execute the Online Activation with the following command:

```
kenaflow.exe --activate <e-mail-address>
```

As `<e-mail-address>`, use the email address specified when purchasing the license. This email address is linked to the serial code.

!! Pay attention to upper and lower case.

Note that the computer requires internet access for Online Activation. You must be able to open the page https://lic.kenaro.com via your browser.

If you can do this, *kenaflow* most likely can also access the licensing interface, unless other settings on the local or corporate network prevent this.

If necessary, you can customize the `kenaflow.exe.config` file to configure the proxy settings.

Find the following section in the file `kenaflow.exe.config`:

```
<!--
    <system.net>
      <defaultProxy enabled="true" useDefaultCredentials="true">
         <proxy proxyaddress="http://myproxyserver:8080" bypassonlocal="True"/>
      </defaultProxy>
    </system.net>
-->
```

This area is commented out. Remove `<!--` and `-->`. Adjust the settings to your circumstances.

Proceed with article *"Service Account"*.

8.4.4.1 Multiple Instances of *kenaflow*

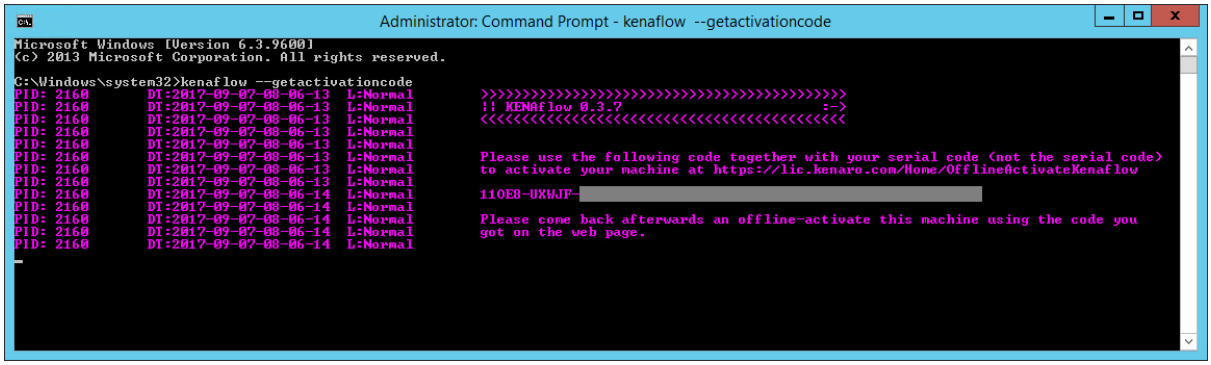
The activation is only done for one instance of *kenaflow* per server! If you have multiple instances on one server and *kenaflow* was already activated on the server there is no need to do it again.

8.4.5 Offline Activation

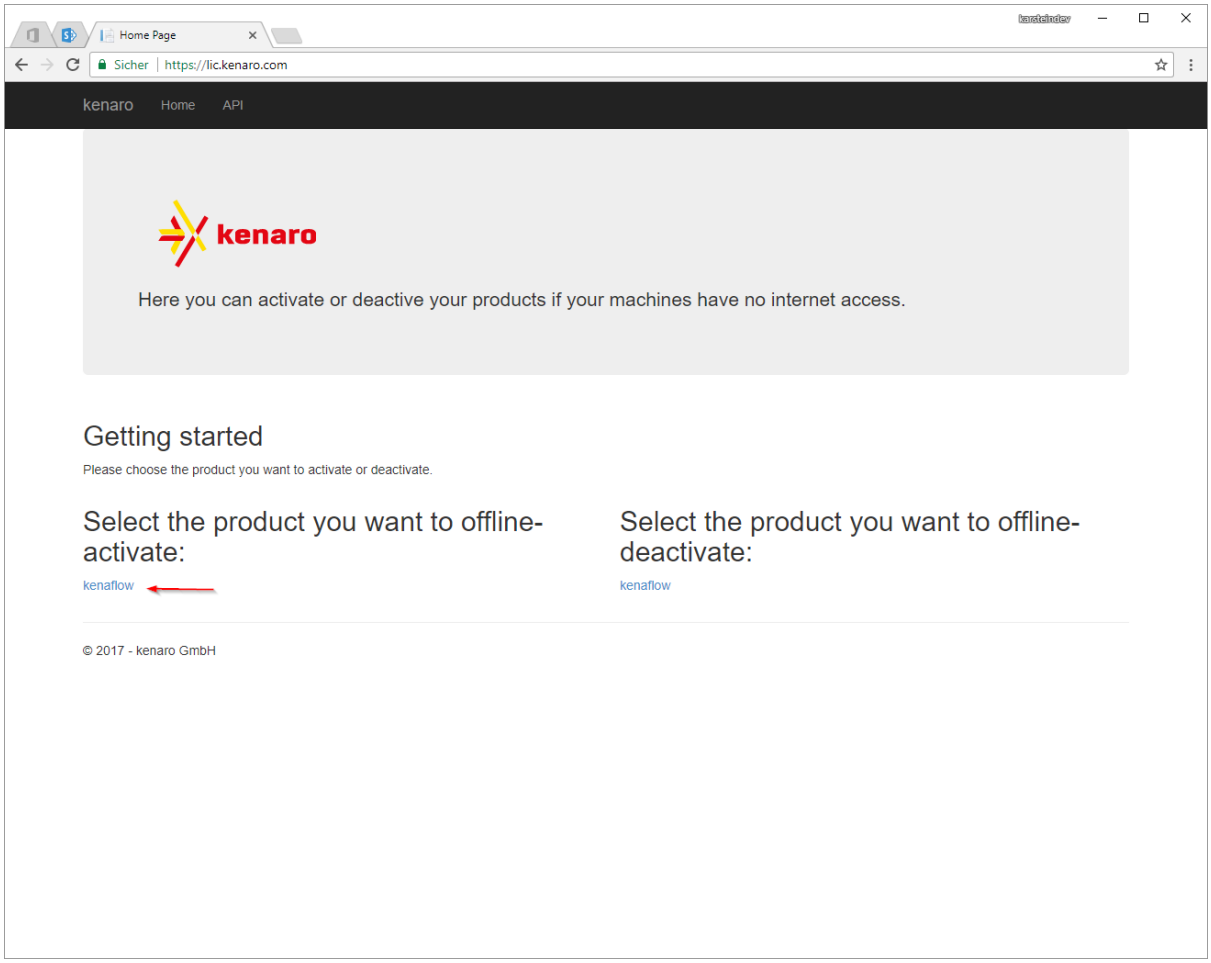
As part of the offline activation, you must now determine the "Activation Code" of the computer. To do this, execute the following command:

```
kenaflow.exe --getactivationcode
```

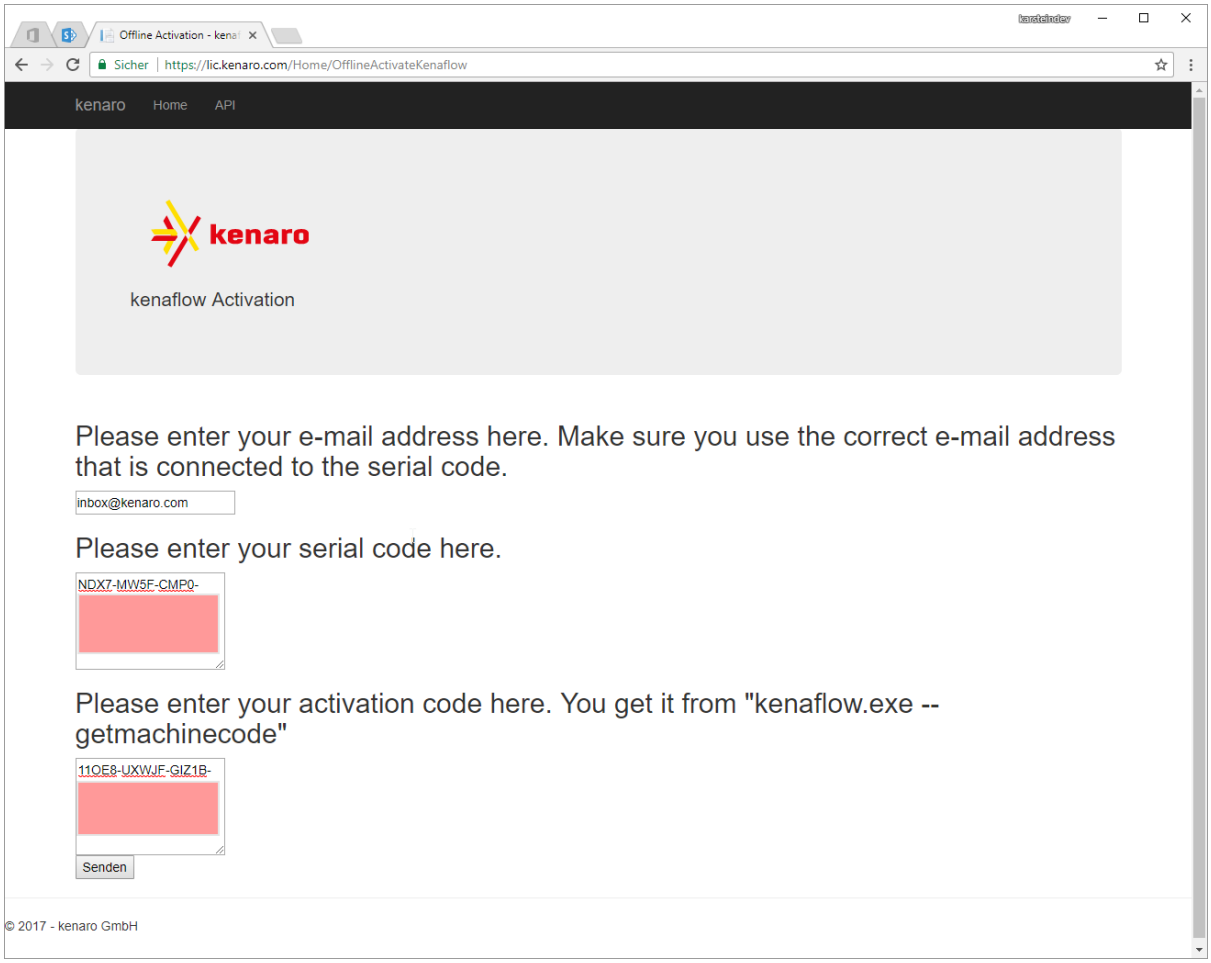
This step is only necessary for a single instance of *kenaflow* on a computer because the activation code for this computer is the same for every instance of *kenaflow*.



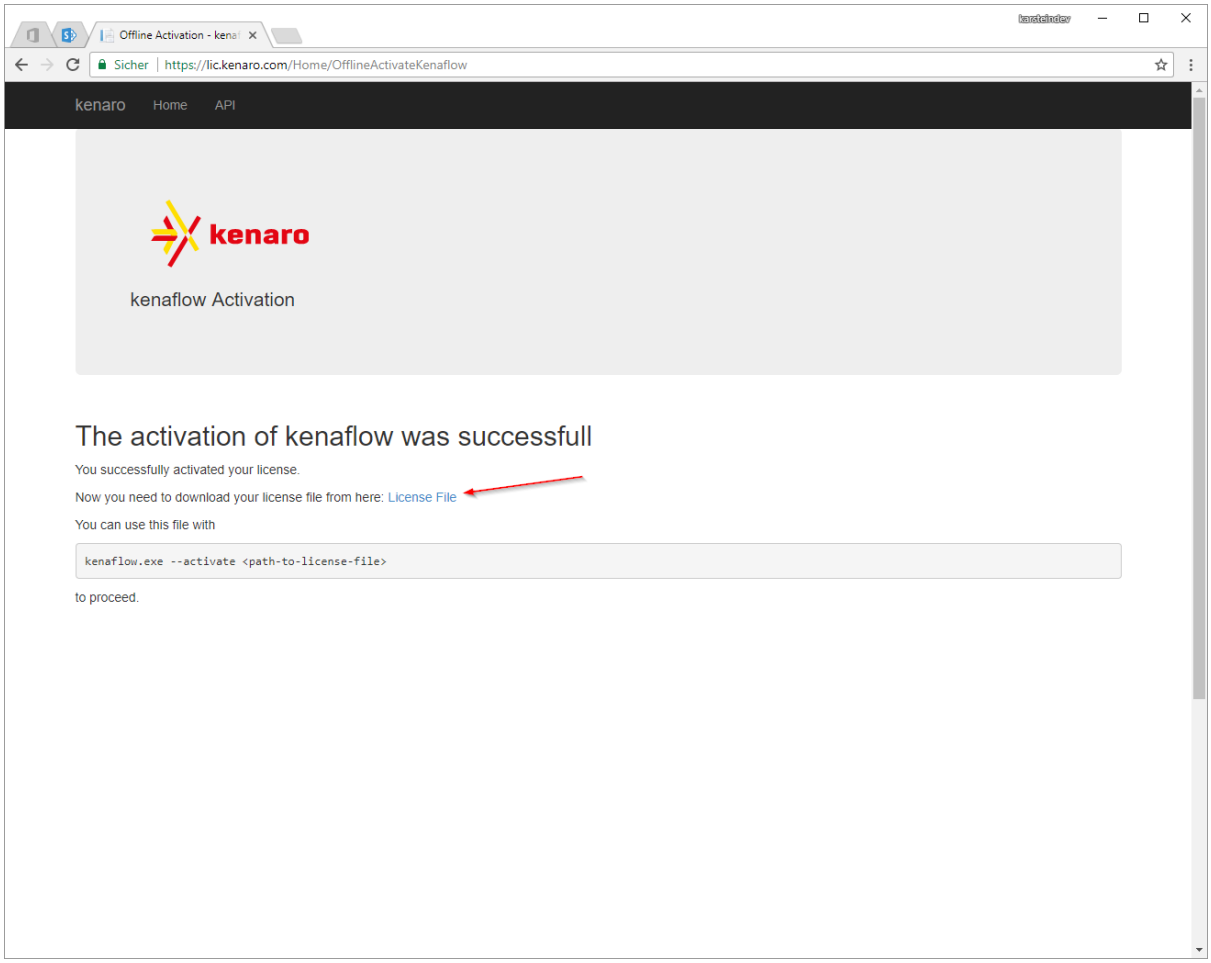
You can activate your *kenaflow* license at https://lic.kenaro.com with your activation code, serial code and the e-mail address you provided when you purchased.



There you have to enter the e-mail address, serial code and activation code.



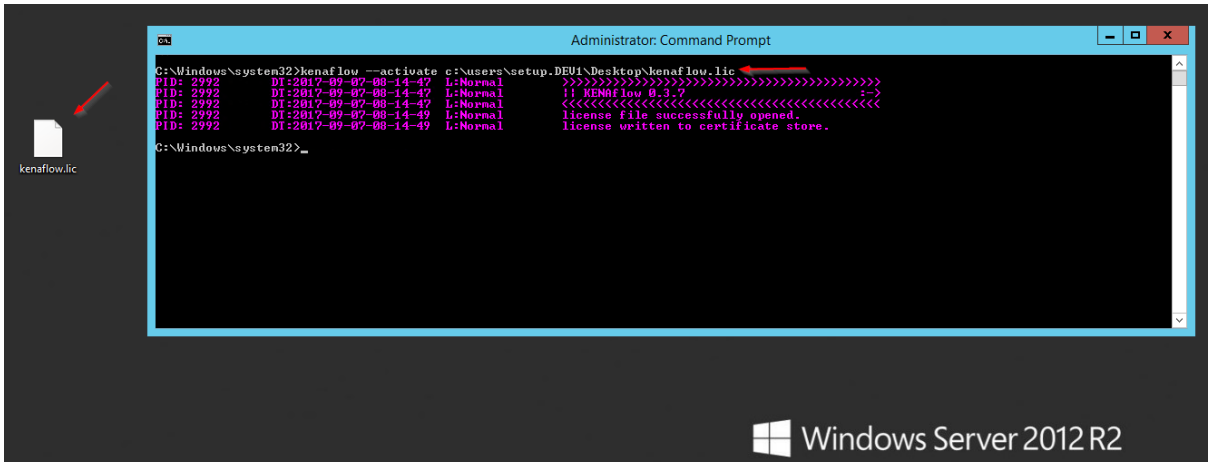
Insofar as your license has not yet been activated on another computer (**without deactivation**) and the serial code has been recognized, you will be offered the license file for download.



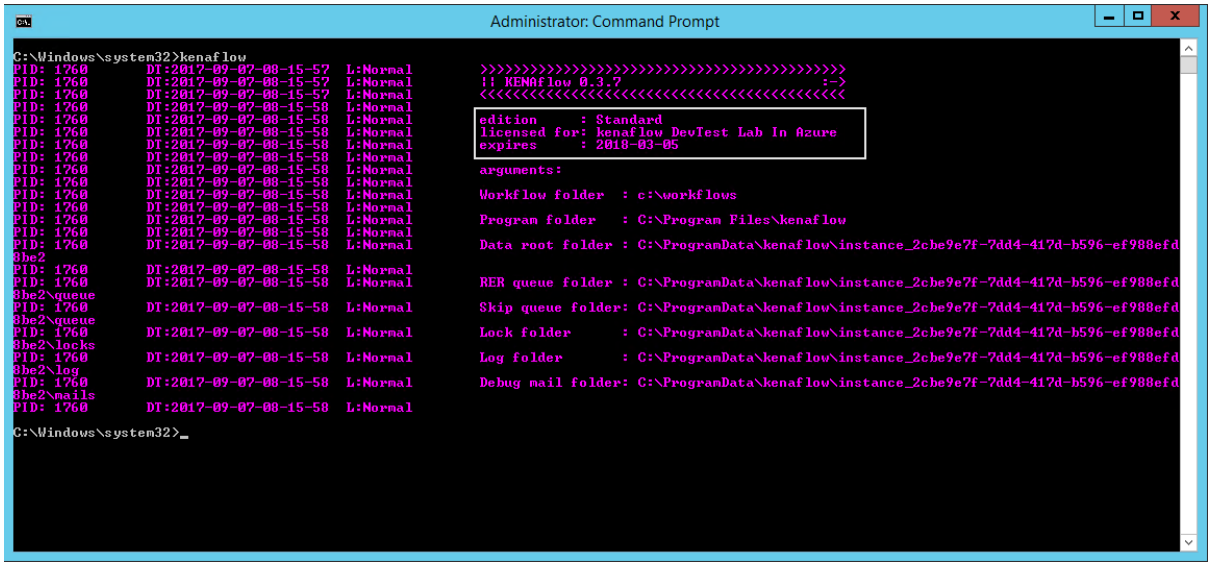
Copy this license file to the *kenaflow* computer and run the command:

```
kenaflow.exe --activate=<license-file>
```

Enter the full path to the license file, which can also be in the *kenaflow* directory.



Then execute the following command for each *kenaflow* instance in the respective program directory (for example C:\Program Files\kenaflow):



Your license has been reviewed and you see the licensing information.

The license file can then be deleted or backed up.

If you activate again via the portal https://lic.kenaro.com for an already activated computer, you will receive the same license file again.

If you re-run activation via the portal https://lic.kenaro.com after a previous deactivation for the same computer, you will receive a new license file.

This step is only necessary for a single *kenaflow* instance on a computer!

8.4.6 Deactivation

Deactivating the license will deactivate all *kenaflow* instances on the computer. Deactivation is not necessary to disable a *kenaflow* instance on a computer.

Deactivation is also possible without direct access to the computer. For this the activation code used during activation is required.

To deactivate, visit this https://lic.kenaro.com/Home/OfflineDeactivateKenaflow web page via your Bowser.

Enter the requested information there. The activation code is identical to the code used at the time of activation.

In parallel to the deactivation on the internet, call upon *kenaflow* with the following command to deactivate the software:

```
kenaflow.exe ---deactivate
```

8.4.7 Service Account

Running *kenaflow* requires a user account - a “functional user” or “service account”.

The account on the executing computer needs to have normal user rights. Most likely it will be a account with only membership in Active folder group “Domain Users”.

It is recommended to use an AD user account whose password does not expire.

However, it is not mandatory to use an AD account.

The account used must have *list*, *read* and *execute* permissions on the program folder of the *kenaflow* instance (for example, `C:\Program Files\kenaflow`).

The account used must have read, create, list, delete and write access on the data folder of the *kenaflow* instance (for example `C:\ProgramData\kenaflow\instance_8a040c56-4343-4f7a-9713-9aa56b1dcc21`).

To set the required permissions please use program mode *`--securing`*.

E.g.:

```
kenaflow.exe --securing --acount "svc.kenaflow"
```

The service account should be used to start the executable file `kenaflow.exe`. In the context of the service account, the program code is executed.

Proceed with *Automatic Execution*.

8.4.8 Automatic Execution

kenaflow is not a Windows service.

kenaflow does not start independently, e.g. *not* when Windows starts.

There are several ways to start *kenaflow* automatically. We prefer to use the "Windows Task Scheduler" that is part of the operating system.

kenaflow provides the program switch `--installschedtask --account <service-account> --password <password>` to create a start task for the *kenaflow* instance in the Task Scheduler. (Details here: *`--installschedtask`*.)

The specified account is used to execute *kenaflow*. The password is required to register the task to run in the background without manual logon.

!! By default this account will also be used to access SharePoint if a SharePoint related workflow has no own configuration for credentials!

Another option is to register *kenaflow* as a Windows service. Therefore, the *kenaflow* program folder contains the program `nssm` (see https://nssm.cc/).

8.4.9 Path Variable

If you are running a single *kenaflow* instance on the computer, it is recommended that you add the *kenaflow* program folder to the `PATH` environment variable of the Windows operating system.

To do this, start a command line (`cmd.exe`) with elevated privileges ('as administrator') and execute the following command:

```
setx PATH "%PATH%;<kenaflow-program-folder>" /M
```

Replace `<kenaflow-program-folder>` with the actual folder.

CHAPTER

NINE

OPERATION

Here we describe common topics related to the operation the the *kenaflow* system.

- *Update*
- *Logging*
- *Monitoring*

9.1 Update

kenaflow should be updated regularly.

Notifications about new versions of *kenaflow* are available on the website: https://kenaflow.com.

In the following sections, we will show you,

1. how to update *kenaflow* online, on a computer with an existing internet connection,
2. or how to update *kenaflow* offline, on a computer without an existing internet connection,
3. which *typical solutions* we can provide for known update problems and
4. what *hints* about the update process we want you to take along.

During the installation command line switch `--configfile <file>` can be used to install a prepared configuration file.

You can specify switch `--noeditconfig` to suppress the final call of the editor for the configuration file.

When updating, you can specify `-u` and `-d <directory>` to be able to specify the target directory for an offline update.

If an update is attempted with the same program version, the update will fail. If the same version is to be installed again, `--force` can be used.

If `--nongen` is used, the final generation of the "Native Images" is not performed. This is a task that should improve the performance of the engine.

Before the update a copy of the file kenaflow.config is backuped. This is named after the old version and with an additional number, so that unique file names are always created, e.g. kenaflow.conf.4.0.0.0-1.bak.

During the update, all files are deleted from the kenaflow program directory, with the exception of

- kenaflow.conf*
- .update-pending
- kenaflow.install.exe.config
- kenaflow.exe.config
- custom-library.ps1
- *.old

9.1.1 Online update

To update *kenaflow* you have to start a command line (`cmd.exe`) with elevated privileges ("as administrator ").

Do not start the setup from a PowerShell command line because the installer will search for running powershell.exe processes.

Close the following programs that interact with *kenaflow*:

- ULSViewer (`ulsviewer.exe`)
- PowerShell (`powershell.exe`)
- PowerShell ISE (`powershell_ise.exe`)
- Visual Studio Code (`code.exe`)

Also, make sure that no other user who was or is logged on to the *kenaflow* computer is currently running these processes. You can check this via PowerShell or via the Windows Task Manager.

Now execute the following steps:

1. Change to the program directory of *kenaflow*, e.g. `C:\Program Files\kenaflow`
2. Start `kenaflow.install.exe -u`
3. The update process starts. This can take a while. Before binaries can be installed the updated *kenaflow* instance need to shutdown. This happens automatically.
4. After setup *kenaflow* will not start automatically by itself. But if you configured *Automatic Execution* it will (re)start as configured.

Please read the artice *Typical Solutions*.

9.1.2 Offline update

To perform an offline update, you must first download the setup and copy it to the *kenaflow* computer.

Here you will find the setup:

https://kenaflow.blob.core.windows.net/release-3-0/kenaflow.latest.zip

Extract the ZIP file to any folder.

Close the following programs that interact with *kenaflow*:

- ULSViewer (`ulsviewer.exe`)
- PowerShell (`powershell.exe`)
- PowerShell ISE (`powershell_ise.exe`)
- Visual Studio Code (`code.exe`)

Also, make sure that **no other user** who was or is logged on to the *kenaflow* computer is currently running these processes. You can check this via PowerShell or via the Windows Task Manager.

Open a command line (`cmd.exe`) with elevated privileges ('Run as Administrator') and change to the directory to which you have unzipped the ZIP. **Do not start the setup from a PowerShell command line because the installer will search for running powershell.exe processes.**

In the command line, start the following command and replace `<target folder>` with the directory containing the *kenaflow* instance to be updated.

```
kenaflow.install.exe --installupdate -d "<target-folder>"
```

1. The update process starts. This can take a while. Before binaries can be installed the updated *kenaflow* instance need to shutdown. This happens automatically.

The preferd way is to create a sub folder in the *kenaflow* program folder named `update-<new-version-number>` where `<new-version-number>` is the number of the next kenaflow version. This number looks like this: "4.0.1". - Unzip the downloaded update file to this folder. Then open `cmd.exe` ("as Administrator") and switch to the update folder. Then simply run `kenaflow.install.exe --installupdate`.

After setup *kenaflow* will not start automatically by itself. But if you configured *Automatic Execution* it will (re)start as configured.

Please read the artice *Typical Solutions*.

9.1.3 Typical Solutions

If you need a proxy for the internet connection in the company, then modify the file `kenaflow.install.exe.config`.

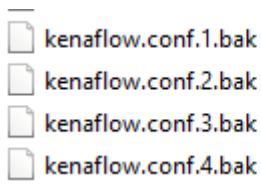
```
    <!--
            <system.net>
                    <defaultProxy enabled="true" useDefaultCredentials="true">
                            <proxy proxyaddress=http://myproxyserver:8080␣
↪bypassonlocal="True"/>
                    </defaultProxy>
            </system.net>
    -->
```

This area is commented out. Remove `<!--` and `-->`. Adjust the settings to your circumstances.

This file will not be overwritten during an update!

If the update aborts:

- In the Task Manager of Windows, check if the processes `powershell.exe` or `powershell_ise.exe` are active. It is possible that these could not be terminated correctly. Close the processes manually and start the update again.
- Also, check if Visual Studio Code still has active processes. These are called `code.exe`. If necessary, stop them.
- Restart the machine. Maybe another process has locked file access to *kenaflow* files.
- Change to the program folder of the *kenaflow* instance, e.g. `C:\Program Files\kenaflow`. Search for the global configuration file `kenaflow.conf`. You may see several files, e.g.:

  Check the backup file with the highest number. Was it created at the time of the failed update attempt?

  If so, manually **backup** and delete the current `kenaflow.conf`, copy the backup file with the highest number, e.g. `kenaflow.conf.4.0.0-1.bak` and rename the copied file to `kenaflow.conf`.

- If not, save the program folder, especially your `custom-library.ps1` and `kenaflow.conf` (with all backups of this file). Then move the program folder to another location. Then install the version of *kenaflow* that matches the current version of `kenaflow.conf`. Then copy back the backed up files `kenaflow.conf` and all backups, as well as `custom-library.ps1`. Then run the update normally or with the method described above.
- If you could not move the program folder in the step described above, then some process in Windows has opened a file. If necessary, restart the computer.

9.1.4 Hints

We would like to give you the following information and background information for the update.

Before the update, the update program creates a file ".update-pending" in the program folder so that all running *kenaflow* processes are informed and stopped.

kenaflow will not start automatically after the update, if not scheduled as a "Scheduled Task" or Windows service. This depends on your system configuration. Please read article "*Automatic Execution*".

During the update, the global configuration file `kenaflow.conf` is saved as `kenaflow.conf.<Old-program-version>.<X>.bak`

- `<old-program-version>` is the version number of *kenaflow* that is currently being updated.
- `<x>` is a serial number that starts at 1 and is used to create unique file names.

The existing settings of this file are transferred to the updated version of the `kenaflow.conf` file. Any newly added settings will be left as default. After updating, check the `kenaflow.conf` file by running the `kenaflow.exe --editconfig` command.

During the update, all files in the program folder are deleted.

Except:

- kenaflow.conf
- kenaflow.conf.*.bak
- custom-lib.ps1
- kenaflow.install.exe.config
- kenaflow.exe.config
- .update-pending

Please note that you are not allowed to save your own files in the program folder of *kenaflow*!

kenaflow is constantly being further developed. When executing updates with the command `kenaflow.install -u`, however, only updates within the same version, e.g. from version 4.0.**01** to 4.0.**02**.

Currently we see no reason for using "minor version updates" such as 4.**1**.x.

If a new major version of *kenaflow* is released - e.g. **5**.0.x - it is necessary to perform additional steps before updating. This has the background that we can only guarantee 100% compatibility of the workflows with the new version, of *kenaflow* within the same version. This is due in particular to the fact that we use external components, some of which change. With each new major version we update these components.

An example: In the SharePoint PnP PowerShell module used by *kenaflow*, there was an `Execute-PnPQuery` cmdlet. This was renamed to `Invoke-PnPQuery` by the developers of the framework. In the version of *kenaflow* 1.0.x we are still using a version of the SharePoint PnP PowerShell module that contains the old name `Execute-PnPQuery`. Beginning with version 2.0.x of *kenaflow* uses a newer version of the framework, which uses the cmdlet called `Invoke-PnPQuery`.

We recommend that you install the new version in a separate folder on the *kenaflow* server so that you can make such changes and similar adjustments for each workflow. Different versions of *kenaflow* can be run side by side. See *Architecture*.

The `Version` property must be adjusted in the workflow configuration. It must correspond to the *kenaflow* version.

Workflows can be moved from the workflow root folder of the old version to the workflow root folder of the new *kenaflow* version. There, the `Version` property in the `_wfconfig.ps1` workflow configuration must be updated manually by the workflow developer.

However, if you want to update an existing installation, carry out the following steps:

1. Edit the global configuration with `kenaflow.exe --editconfig`.
2. Find the "Branch" property in the editor.

3. Replace `Release-2-0` with `Release-3-0`.
4. Save the file and close the editor.
5. Perform an update: `kenaflow.install -u`.

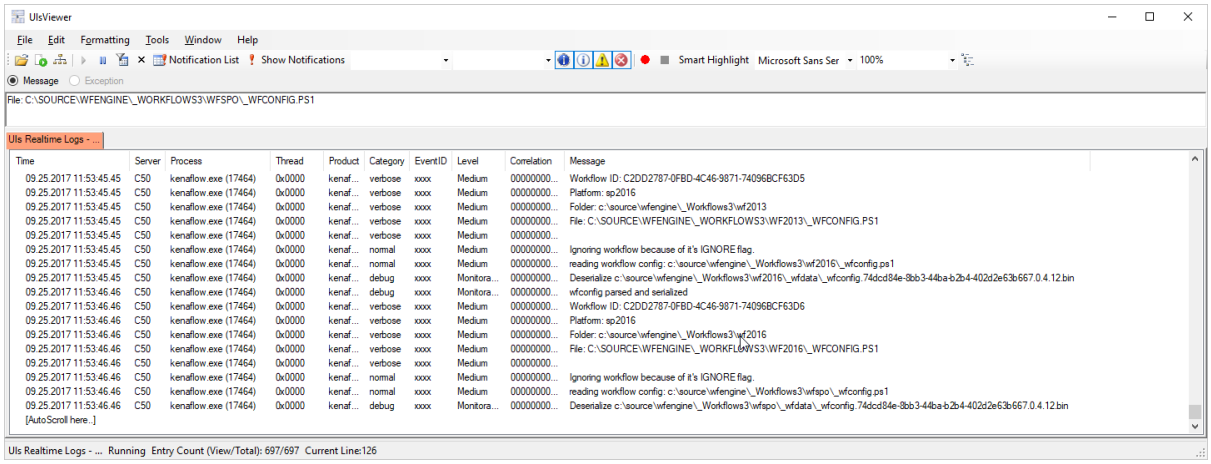
Then the existing installation was replaced by the new version.

9.2 Logging

Each *kenaflow* instance has a `logDir` setting in the global configuration (section *"Logging"*.

This specifies the directory in which log files are stored.

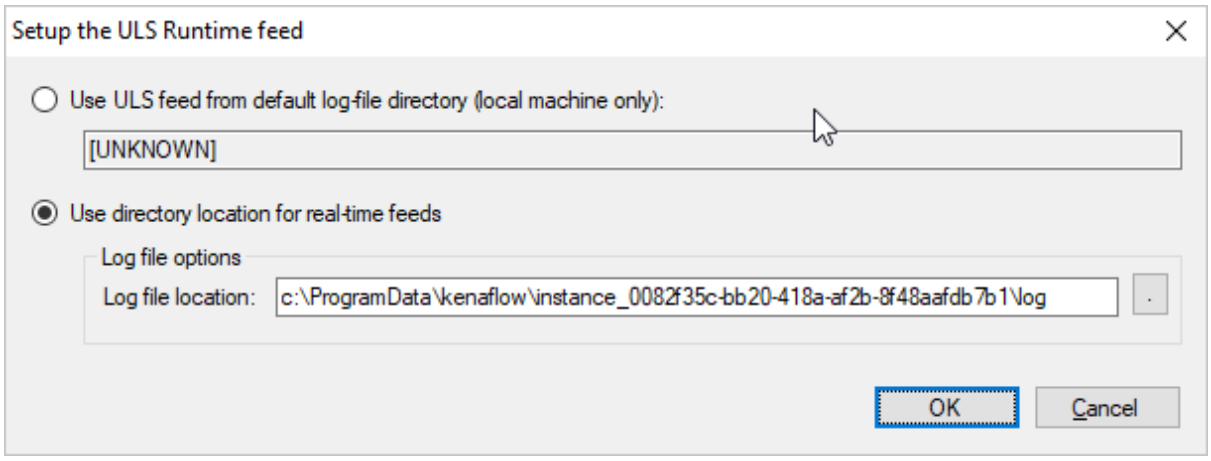
The log files of *kenaflow* correspond to the log file format of SharePoint. The tool "ULSViewer" from Microsoft can be used to analyse the logs.



The ULSViewer can be downloaded here:

https://www.microsoft.com/en-us/download/details.aspx?id=44020

The menu for the "Live Logs" opens with `[Strg]+[U]`:



Enter the log directory here. By default its in `C:\ProgramData\kenaflow\<instance_id>\log`.

9.2.1 Format

`Timestamp` <TAB> character `Process` <TAB> character `TID` <TAB> character `Area` <TAB> character `Category` <TAB> character `EventID` <TAB> character `Level` <TAB> character `Message` <TAB> character `Correlation`

9.2.1.1 Timestamp

Date and time of the log event.

!! The ULSViewer has a small bug. On systems with a cultur other then "en-US" and/or a different time format it shows first the month before the day in the time stamp. That's valid for "en-US" but not for german dates. E.g. it shows "12.31.2018" instead of "31.12.2018".

9.2.1.2 Process

It's always "kenaflow.exe" followed by the Windows process ID.

9.2.1.3 TID

kenaflow makes heavy use of theads. Each created thread gets an ID that is unique inside the current *kenaflow* process.

9.2.1.4 Area

Always "kenaflow".

9.2.1.5 Category

Thats the *kenaflow* internal log message severity: Debug, Verbose, Normal, Warning, Error, Critical. It's slightly different to the SharePoint log levels (see section "Levels" below).

9.2.1.6 EventID

The `EventID` indicates the *type* of the kenaflow (sub) process that has created the log message.

| EventID | (Sub) Process | | - | - | | kflw | kenaflow process - just started. | | exe3 | Workflow Runner process for SharePoint 2013 | | exe6 | Workflow Runner process for SharePoint 2016 | | exe9 | Workflow Runner process for SharePoint 2019 | | exeo | Workflow Runner process for SharePoint Online | | exep | Workflow Runner process for PowerShell | | rer3 | Remote Event Runner process for SharePoint 2013 | | rer6 | Remote Event Runner process for SharePoint 2016 | | rer9 | Remote Event Runner process for SharePoint 2019 | | rero | Remote Event Runner process for SharePoint Online | | rerp | Remote Event Runner process for PowerShell | | rerg | Global Remote Event Receiver | | asmf | Add State Field | | bacl | Backup Config List | | cuea | Cleanup ALL Error Lists | | cue3 | Cleanup Error List for SharePoint 2013 | | cue6 | Cleanup Error List for SharePoint 2016 | | cue9 | Cleanup Error List for SharePoint 2019 | | cueo | Cleanup Error List for SharePoint Online | | cclm | Create Config List | | cdlm | Create Data List | | celm | Create Error List | | chlm | Create History List | | cslm | Create State List | | cpsm | Create Permission Sets | | shcl | Show/Hide Config List | | shdl | Show/Hide Data List | | shel | Show/Hide Error List | | shhl | Show/Hide History List | | shsl | Show/Hide State List | | cpwf | Copy Workflow | | dump | Dump remote event | | crpc | Create Performace Counters | | main | "Main" runner |

(to be continued)

9.2.1.7 Level

Possible Values:

- Critical
- Unexpected
- High
- Medium
- VerboseEx

9.2.1.8 Message

If a message is longer than xxx chacters the line is seperated into two or more lines. In this case message is split and `...` is added at the end of a line (that has following lines) and also `...` is added to the next line.

9.2.1.9 Correlation

This contains an empty GUID if the engine has no certain execution context.

As soon as log messages are related to a certain workflow this information contains the unique workflow id.

9.2.2 Sample

```
   Timestamp                        Process                                        ␣
↪        TID           Area                                   Category             ␣
↪                  EventID        Level              Message         Correlation
   01/22/2019 14:52:01.623         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
↪>>>>>>>>>>>        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:01.623         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               || KENAflow 2.0.25               ␣
↪        :->        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:01.623         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
↪<<<<<<<<<<<        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.133         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               edition     :␣
↪Standard        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.133         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               licensed for: development //␣
↪kenaro        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.133         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               expires     : 2019-12-
↪31        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.133         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               license type:␣
↪:Subscription        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.133         kenaflow.exe (22544)                            ␣
↪        0x000C        kenaflow                               normal               ␣
↪                  kflw        High               arguments: --exec --workflow C:\
↪Source\wfengine\_workflows\loop-interrupt-test\test-wf        00000000-0000-0000-
↪0000-000000000000
```

(continues on next page)

(continued from previous page)

```
   01/22/2019 14:52:03.133         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Workflow folder  : C:\Source\
↪wfengine\_workflows\_workflowsDemo        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.133         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Program folder   : C:\Source\
↪wfengine\bin1\Debug        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Data root folder : C:\
↪ProgramData\kenaflow\instance_74dcd84e-8bb3-44ba-b2b4-
↪402d2e63b667        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               RER queue folder : C:\Source\
↪wfengine\_wfexecdata\queue        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Skip queue folder: C:\Source\
↪wfengine\_wfexecdata\skip        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Lock folder      : C:\Source\
↪wfengine\_wfexecdata\lock        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Log folder       : C:\Source\
↪wfengine\_wfexecdata\log        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Debug mail folder: C:\Source\
↪wfengine\_wfexecdata\mails        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              normal                ␣
↪                 kflw        High               Current user: C61\
↪user1        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              verbose               ␣
↪                 kflw        Medium             reading workflow config␣
↪file        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.237         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              debug                 ␣
↪                 kflw        VerboseEx          Deserialize C:\Source\wfengine\_
↪workflows\loop-interrupt-test\test-wf\_wfdata\_wfconfig.74dcd84e-8bb3-44ba-b2b4-
↪402d2e63b667.2.0.25.bin        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.743         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              debug                 ␣
↪                 kflw        VerboseEx          deserialized wfconfig␣
↪used:        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.743         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              verbose               ␣
↪                 kflw        Medium             Workflow ID: 23095FC3-A8EA-4EF1-
↪AFE5-979B3A91E128        00000000-0000-0000-0000-000000000000
   01/22/2019 14:52:03.743         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              verbose               ␣
↪                 kflw        Medium             Platform: sp2016        00000000-
↪0000-0000-0000-000000000000
   01/22/2019 14:52:03.743         kenaflow.exe (22544)                    ␣
↪        0x000C        kenaflow                              verbose               ␣
↪                 kflw        Medium             Folder: C:\Source\wfengine\_
↪workflows\loop-interrupt-test\test-wf        00000000-0000-0000-0000-000000000000
```

(continues on next page)

(continued from previous page)

```
    01/22/2019 14:52:03.743         kenaflow.exe (22544)
↪         0x000C        kenaflow                                verbose
↪                 kflw        Medium              File: C:\SOURCE\WFENGINE\_
↪WORKFLOWS\LOOP-INTERRUPT-TEST\TEST-WF\_WFCONFIG.PS1        00000000-0000-0000-
↪0000-000000000000
    01/22/2019 14:52:03.743         kenaflow.exe (22544)
↪         0x000C        kenaflow                                verbose
↪                 kflw        Medium              Web: https://intranet16.
↪sharepoint.farm/sites/kenaflow        00000000-0000-0000-0000-000000000000
    01/22/2019 14:52:03.743         kenaflow.exe (22544)
↪         0x000C        kenaflow                                verbose
↪                 kflw        Medium              List: TestList        00000000-
↪0000-0000-0000-000000000000
    01/22/2019 14:52:03.743         kenaflow.exe (22544)
↪         0x000C        kenaflow                                verbose
↪                 rqbm        Medium              IPC GlobalStop event created!
↪         d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.743         kenaflow.exe (22544)
↪         0x000C        kenaflow                                normal
↪                 rqbm        High                initialize mode 'ExecuteWorkflow
↪'         d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                normal
↪                 exe6        High                reading workflow config file: C:\
↪Source\wfengine\_workflows\loop-interrupt-test\test-wf\_wfconfig.
↪ps1        d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                debug
↪                 exe6        VerboseEx           Deserialize C:\Source\wfengine\_
↪workflows\loop-interrupt-test\test-wf\_wfdata\_wfconfig.74dcd84e-8bb3-44ba-b2b4-
↪402d2e63b667.2.0.25.bin        d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                debug
↪                 exe6        VerboseEx           deserialized wfconfig
↪used:        d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                verbose
↪                 exe6        Medium              Workflow ID: 23095FC3-A8EA-4EF1-
↪AFE5-979B3A91E128        d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                verbose
↪                 exe6        Medium              Platform: sp2016        d5eb636a-
↪f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                verbose
↪                 exe6        Medium              Folder: C:\Source\wfengine\_
↪workflows\loop-interrupt-test\test-wf        d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                verbose
↪                 exe6        Medium              File: C:\SOURCE\WFENGINE\_
↪WORKFLOWS\LOOP-INTERRUPT-TEST\TEST-WF\_WFCONFIG.PS1        d5eb636a-f174-4b66-
↪bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                verbose
↪                 exe6        Medium              Web: https://intranet16.
↪sharepoint.farm/sites/kenaflow        d5eb636a-f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                verbose
↪                 exe6        Medium              List: TestList        d5eb636a-
↪f174-4b66-bb53-360a77d0fe85
    01/22/2019 14:52:03.858         kenaflow.exe (22544)
↪         0x0012        kenaflow                                normal
↪                 exe6        High                workflow
↪execution        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
   01/22/2019 14:52:03.858         kenaflow.exe (22544)                    ␣
↪        0x0012        kenaflow                              normal          ␣
↪                 exe6        High               running in slot:        23095fc3-
↪a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:03.858         kenaflow.exe (22544)                    ␣
↪        0x0012        kenaflow                              verbose         ␣
↪                 exe6        Medium             current client is configured NOT␣
↪to send log messages to client        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:04.518         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              debug           ␣
↪                 exe6        VerboseEx          Workflow lock file : C:\Source\
↪wfengine\_wfexecdata\lock\23095FC3-A8EA-4EF1-AFE5-979B3A91E128.
↪lock        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:04.518         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              verbose         ␣
↪                 exe6        Medium             workflow locked        23095fc3-
↪a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:05.049         kenaflow.exe (22544)                    ␣
↪        0x0018        kenaflow                              verbose         ␣
↪                 exe6        Medium             Last Run: 2019-01-
↪22T13:34:31Z        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:05.049         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal          ␣
↪                 exe6        High               connecting to SharePoint...
↪        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:05.049         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal          ␣
↪                 exe6        High               Checking RER handler␣
↪settings        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:08.207         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug           ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:08.207         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              normal          ␣
↪                 exe6        High               workflow config 'C:\SOURCE\
↪WFENGINE\_WORKFLOWS\_WORKFLOWSDEMO\SPO_LEVEL1\_WFCONFIG.PS1'␣
↪ADDING        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:08.207         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              normal          ␣
↪                 exe6        High               workflow config 'C:\SOURCE\
↪WFENGINE\_WORKFLOWS\_WORKFLOWSDEMO\SP2013_LEVEL1\_WFCONFIG.PS1'␣
↪ADDING        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:08.207         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              normal          ␣
↪                 exe6        High               workflow config 'C:\SOURCE\
↪WFENGINE\_WORKFLOWS\_WORKFLOWSDEMO\SP2013_LEVEL2\_WFCONFIG.PS1'␣
↪ADDING        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:08.207         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              normal          ␣
↪                 exe6        High               workflow config 'C:\SOURCE\
↪WFENGINE\_WORKFLOWS\_WORKFLOWSDEMO\SP2016_LEVEL1\_WFCONFIG.PS1'␣
↪ADDING        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:13.156         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug           ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:18.238         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug           ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
   01/22/2019 14:52:22.070         kenaflow.exe (22544)                    ␣
↪        0x0014        kenaflow                              warning          ␣
↪                 exe6        Unexpected         WebHook ping checker #0 failed 6␣
↪times and changed to FAILURE state!        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:23.178         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug            ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:28.232         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug            ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:28.936         kenaflow.exe (22544)                    ␣
↪        0x001A        kenaflow                              verbose          ␣
↪                 exe6        Medium             Check for RER url: https://
↪kenaflow11.sharepoint.farm/rer/23095FC3-A8EA-4EF1-AFE5-979B3A91E128/
↪        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:28.936         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal           ␣
↪                 exe6        High               creating PowerShell␣
↪session        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:29.251         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug            ␣
↪                 exe6        VerboseEx          SharePointPnP.PowerShell.
↪Commands.Base.SPOnlineConnection        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.072         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              verbose          ␣
↪                 exe6        Medium             Web Id: 14a9e54e-e43c-4e47-9fe3-
↪9d4be1283e42        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.072         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              verbose          ␣
↪                 exe6        Medium             connection to workflow error␣
↪list        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.374         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal           ␣
↪                 exe6        High               current user: i:0#.w|sharepoint\
↪kenaflow (Microsoft.SharePoint.Client.UserIdInfo)        23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
   01/22/2019 14:52:30.374         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal           ␣
↪                 exe6        High               connected        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.374         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal           ␣
↪                 exe6        High               Loading config values...
↪        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.475         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal           ␣
↪                 exe6        High               Last action timestamp not␣
↪found        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.475         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              normal           ␣
↪                 exe6        High               Set default config␣
↪variables        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.777         kenaflow.exe (22544)                    ␣
↪        0x0016        kenaflow                              verbose          ␣
↪                 exe6        Medium             List Id: 9472ad43-c8f0-4d29-8949-
↪c2776f628025        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.777         kenaflow.exe (22544)                    ␣
↪        0x001C        kenaflow                              normal           ␣
↪                 exe6        High               Preparing query...
↪        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
   01/22/2019 14:52:30.878         kenaflow.exe (22544)                    ␣
↪        0x001C        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          <View><Query><Where></Where>
↪<OrderBy><FieldRef Name="ID" /></OrderBy></Query><RowLimit>1000</RowLimit></View>
↪        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.878         kenaflow.exe (22544)                    ␣
↪        0x001C        kenaflow                              verbose              ␣
↪                 exe6        Medium             executing query...
↪        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:30.979         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              verbose              ␣
↪                 exe6        Medium             got 1 item(s)        23095fc3-
↪a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.080         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              normal               ␣
↪                 exe6        High               current item: #1 // 32914150-
↪08eb-4018-b91d-9326a163252c        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.080         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          Current 'editor':␣
↪ingo        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.080         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          Current 'last modified': 2019-01-
↪22T13:41:24Z        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.080         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          Item lock file: C:\Source\
↪wfengine\_wfexecdata\lock\14a9e54ee43c4e479fe39d4be1283e42\
↪9472ad43c8f04d298949c2776f628025\3291415008eb4018b91d9326a163252c.
↪lock        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.382         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              verbose              ␣
↪                 exe6        Medium             item locked        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'SetKFScriptRunTimeout'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          END CMDLET 'SetKFScriptRunTimeout
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'GetKFScriptRunTimeout'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          END CMDLET 'GetKFScriptRunTimeout
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'GetKFScriptRunTimeout'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          END CMDLET 'GetKFScriptRunTimeout
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          BEGIN CMDLET 'Write_KFLog
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0x0022        kenaflow                              normal              ␣
↪                 exe6        High               Test-List-WF: 4999s of␣
↪5000s        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:31.586         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          END CMDLET 'Write_KFLog
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:33.199         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:33.199         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              01/22/2019 14:52:34.
↪609         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'ConvertToKFISODateTime'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.609         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          END CMDLET
↪'ConvertToKFISODateTime'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.710         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'GetKFScriptRunTimeout'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.710         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          END CMDLET 'GetKFScriptRunTimeout
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.710         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'GetKFScriptRunTimeout'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.710         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          END CMDLET 'GetKFScriptRunTimeout
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.710         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          BEGIN CMDLET 'Write_KFLog
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.710         kenaflow.exe (22544)                    ␣
↪        0x0026        kenaflow                              normal              ␣
↪                 exe6        High               Test-List-WF: 4996s of␣
↪5000s        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:34.710         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          END CMDLET 'Write_KFLog
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.738         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'ConvertToKFISODateTime'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.738         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          END CMDLET
↪'ConvertToKFISODateTime'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.839         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug               ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'GetKFScriptRunTimeout'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
   01/22/2019 14:52:37.839         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          END CMDLET 'GetKFScriptRunTimeout
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.839         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'GetKFScriptRunTimeout'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.839         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          END CMDLET 'GetKFScriptRunTimeout
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.839         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          BEGIN CMDLET 'Write_KFLog
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.839         kenaflow.exe (22544)                    ␣
↪        0x002A        kenaflow                              normal               ␣
↪                 exe6        High               Test-List-WF: 4993s of␣
↪5000s        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:37.839         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          END CMDLET 'Write_KFLog
↪'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:38.242         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:38.242         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              logfileonly          ␣
↪                 exe6        VerboseEx          workflow folder 'C:\SOURCE\
↪WFENGINE\_WORKFLOWS\_WORKFLOWSDEMO\SP2016_LEVEL1' hash: |_initialize.ps12018-07-
↪13T14:44:58Z|_install.ps12018-07-13T14:44:58Z|_wfconfig.ps12018-08-
↪16T20:31:11Z|preQueryProcessing.ps12018-07-13T14:44:58Z|S_(empty).ps12018-07-
↪13T14:44:58Z|S_0.0.ps12018-07-13T14:44:58Z|S_1.0.ps12018-07-13T14:44:58Z|S_1.1.
↪ps12018-07-13T14:44:58Z|S_2.0.ps12018-07-13T14:44:58Z|S_2.1.ps12018-07-
↪13T14:44:58Z|S_3.0.ps12018-07-13T14:44:58Z|S_4.0.ps12018-07-
↪13T14:44:58Z        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:40.867         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          BEGIN CMDLET
↪'ConvertToKFISODateTime'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:40.867         kenaflow.exe (22544)                    ␣
↪        0xFFFF        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          END CMDLET
↪'ConvertToKFISODateTime'        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:41.573         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:52:59.238         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:53:01.394         kenaflow.exe (22544)                    ␣
↪        0x000A        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          GC collect        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:53:01.394         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              debug                ␣
↪                 exe6        VerboseEx          item execution finished.␣
↪housekeeping.        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
    01/22/2019 14:53:01.519         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 100 seconds before retrying (1 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:01.721         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 120 seconds before retrying (2 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:01.921         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 144 seconds before retrying (3 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:02.233         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 172,8 seconds before retrying (4 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:02.434         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 207,36 seconds before retrying (5 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:02.840         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 248,832 seconds before retrying (6 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:03.142         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 298,5984 seconds before retrying (7 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:03.621         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 358,31808 seconds before retrying (8 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:04.024         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 429,981696 seconds before retrying (9 of 20).       23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:04.225         kenaflow.exe (22544)
↪         0x0010        kenaflow                                debug
↪                 exe6        VerboseEx          workflow root folder␣
↪scan       23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:04.630         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 515,9780352 seconds before retrying (10 of 20).       23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
    01/22/2019 14:53:05.134         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 619,17364224 seconds before retrying (11 of 20).       23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
    01/22/2019 14:53:05.837         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 743,008370688 seconds before retrying (12 of 20).       23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
    01/22/2019 14:53:06.690         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 891,6100448256 seconds before retrying (13 of 20).        23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
    01/22/2019 14:53:07.697         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1069,93205379072 seconds before retrying (14 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
    01/22/2019 14:53:08.805         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1283,91846454886 seconds before retrying (15 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
    01/22/2019 14:53:09.208         kenaflow.exe (22544)
↪         0x0010        kenaflow                                debug
↪                  exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:10.276         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1540,70215745864 seconds before retrying (16 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
    01/22/2019 14:53:11.898         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1848,84258895036 seconds before retrying (17 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
    01/22/2019 14:53:13.854         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 2218,61110674044 seconds before retrying (18 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
    01/22/2019 14:53:14.156         kenaflow.exe (22544)
↪         0x0010        kenaflow                                debug
↪                  exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:16.086         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 2662,33332808852 seconds before retrying (19 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
    01/22/2019 14:53:18.860         kenaflow.exe (22544)
↪         0x001D        kenaflow                                warning
↪                  exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 3194,79999370623 seconds before retrying (20 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
    01/22/2019 14:53:19.167         kenaflow.exe (22544)
↪         0x0010        kenaflow                                debug
↪                  exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:22.136         kenaflow.exe (22544)
↪         0x001D        kenaflow                                error
↪                  exe6        Critical           Maximum retry attempts 20, have␣
↪been attempted.        23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:24.758         kenaflow.exe (22544)
↪         0x0010        kenaflow                                debug
↪                  exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:24.758         kenaflow.exe (22544)
↪         0x001D        kenaflow                                normal
↪                  exe6        High               Exception handling␣
↪started        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
    01/22/2019 14:53:27.498         kenaflow.exe (22544)
↪         0x001D         kenaflow                                verbose
↪                  exe6         Medium              item lock␣
↪released         23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:27.498         kenaflow.exe (22544)
↪         0x001D         kenaflow                                debug
↪                  exe6         VerboseEx           Item processing time:␣
↪56345ms         23095fc3-a8ea-4ef1-afe5-979b3a91e128
    01/22/2019 14:53:27.749         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 100 seconds before retrying (1 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:27.958         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 120 seconds before retrying (2 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:28.160         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 144 seconds before retrying (3 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:28.361         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 172,8 seconds before retrying (4 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:28.664         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 207,36 seconds before retrying (5 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:28.966         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 248,832 seconds before retrying (6 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:29.386         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 298,5984 seconds before retrying (7 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:29.798         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 358,31808 seconds before retrying (8 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:30.200         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 429,981696 seconds before retrying (9 of 20).         23095fc3-a8ea-4ef1-afe5-
↪979b3a91e128
    01/22/2019 14:53:30.830         kenaflow.exe (22544)
↪         0x001D         kenaflow                                warning
↪                  exe6         Unexpected          CSOM request failed. Sleeping␣
↪for 515,9780352 seconds before retrying (10 of 20).         23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
    01/22/2019 14:53:31.434         kenaflow.exe (22544)
↪         0x0010         kenaflow                                debug
↪                  exe6         VerboseEx           workflow root folder␣
↪scan         23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

(continues on next page)

(continued from previous page)

```
   01/22/2019 14:53:31.736         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 619,17364224 seconds before retrying (11 of 20).        23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
   01/22/2019 14:53:32.441         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 743,008370688 seconds before retrying (12 of 20).        23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
   01/22/2019 14:53:33.247         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 891,6100448256 seconds before retrying (13 of 20).        23095fc3-a8ea-4ef1-
↪afe5-979b3a91e128
   01/22/2019 14:53:34.265         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1069,93205379072 seconds before retrying (14 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:53:35.473         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1283,91846454886 seconds before retrying (15 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:53:36.379         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug            ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
   01/22/2019 14:53:36.782         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1540,70215745864 seconds before retrying (16 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:53:38.552         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 1848,84258895036 seconds before retrying (17 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:53:40.489         kenaflow.exe (22544)                    ␣
↪        0x001D        kenaflow                              warning          ␣
↪                 exe6        Unexpected         CSOM request failed. Sleeping␣
↪for 2218,61110674044 seconds before retrying (18 of 20).        23095fc3-a8ea-
↪4ef1-afe5-979b3a91e128
   01/22/2019 14:53:41.395         kenaflow.exe (22544)                    ␣
↪        0x0010        kenaflow                              debug            ␣
↪                 exe6        VerboseEx          workflow root folder␣
↪scan        23095fc3-a8ea-4ef1-afe5-979b3a91e128
```

9.3 Monitoring

9.3.1 Performance Counter

kenaflow offers some basic performance counters to enable third-party monitoring systems to check *kenaflow* activity.

The use of performance counters is disabled by default but can be enabled in the global configuration, section "Monitoring".

If enabled...

Each version of kenaflow creates its own set of performance counters. The performance counter group name starts with "kenaflow" followed by the product version, such as "4.0.0".

! Performance counter groups of kenaflow versions that are not longer used are not removed automatically. You need to run kenaflow from command line with switch `--removeperformancecounter <version-to-remove>` ! ! Sample: `kenaflow.exe --removeperformancecounter "4.0.0"`

9.3.2 Web Hooks

kenaflow is able to inform third-party monitoring systems about its work.

Therefore it is able to call two types of web hooks.

(to be continued.)

9.3.3 Network Connectivity Check

kenaflow maybe need to check connectivity in order to prevent a flood of error messages if it is not "online", not connected to the intranet or internet or not able to contect SharePoint (on-premises or Online).

Therefore *kenaflow* is able to monitor other network related devices to check its connectivity.

In setting "PingICMP" in global configuration section *"Monitoring"* you can specify zero, one or more devices that have to be available over ICMP ping to infornm *kenaflow* about network connectivity.

One entry consists of this settings, separated by | (pipe character):

- IP address or host name
- time between two ICMP pings (in milliseconds)
- maximum latency (in milliseconds)
- mark this entry (device) as "failed" (= not reachable / disconnected) after this number of failures
- mark this entry (device) as "normal" (= reachable / connected) after this number of successful ping AFTER it's in "failed" state

Sample for two entries:

```
172.16.1.1|5000|1|5|5~domaincontroller2.company.local|5000|1|5|5
```

Furthermore *kenaflow* is able to monitor other network related devices to check its connectivity.

In setting "PingWebHook" in global configuration section *"Monitoring"* you can specify zero, one or more devices that have to be available over HTTP web request to infornm *kenaflow* about network connectivity.

One entry consists of this settings, separated by | (pipe character):

- URL to web resource
- time between two HTTP request (in milliseconds)
- desired HTTP response status code

- mark this entry (device) as “failed” (= not reachable / disconnected) after this number of failures
- mark this entry (device) as “normal” (= reachable / connected) after this number of successful ping AFTER it’s in “failed” state

Sample for two entries:

```
http://ping1.company.local|5000|200|5|5~https://172.16.1.
1|5000|200|5|5~
```

The response of the HTTP request is ignored. Only the HTTP response status code is monitored!

9.4 Installer Command Line

The *kenaflow* installer is located in the program folder, e.g. `C:\Program Files\kenaflow`

9.4.1 Overview

The installer has the following switches:

For setup:

- `--i`
- `--d <folder>`
- `--noeditconfig`
- `--configfile <config-file-path>`

For update:

- `--d <folder>`
- `--u`
- `--installupdates`
- `--force`
- `--kill`

9.4.2 Setup

The setup process is decribed here: *Setup*.

9.4.3 Install Updates

From within the *kenaflow* program folder (e.g. `C:\Program Files\kenaflow`) you can run this command to update kenaflow to the newest version:

```
kenaflow.install.exe -u
```

This will check the current version on the internet with the currently installed version. If you version is the same than the installer will stop.

You can force a re-install like this:

```
kenaflow.install.exe -u --force
```

The setup file (a ZIP file) will be downloaded from the internet url https://kenaflow.blob.core.windows.net/release-x-y/kenaflow.latest.zip

`x-y` means the major and minor version number of *kenaflow*, e.g. `3-0`.

In case your machine has no internet access or you want to download it yourself, than un-zip the setup file, open an elevated prompt in the folder where you extracted the files.

Than run:

```
kenaflow.install.exe --installupdates -d <install-folder>
```

... where `-d <install-folder>` is folder of your current installation.

Also here you can use `--force` to re-install the same version as your current one.

9.4.4 Update Automation

It is possible to automate and schedule the *kenaflow* update.

Therefore you can create (manually) a Windows scheduled task for program `kenaflow.exe` with argument `-u`.

In such a scenario you have to ensure that no user or program is blocking files from the *kenaflow* program folder.

We added the program switch `--kill` to allow *kenaflow* installer to terminate blocking applications without warning.

9.5 Uninstall / Remove *kenaflow*

Here we describe how to remove *kenaflow* from a server. Please follow the steps.

1. Stop kenaflow by stopping the main process, e.g. in Windows Task Scheduler
2. Remote the *kenaflow* scheduled task if it was used.
3. To unregister remote event receivers from SharePoint you need to use `--unregisterrer` for each workflow that uses remote events with SharePoint.
4. Deactivate *kenaflow* (*Deactivation*).
5. Delete the program folder (e.g. `C:\Program Files\kenaflow`)
6. Delete the data folder of the *kenaflow* instance (e.g. `C:\ProgramData\kenaflow\instance_<guid>`)
7. Remove the *kenaflow* instance registry key (e.g. `HKLM\Software\kenaflow\<guid>`)
8. Remove the *kenaflow* licensing certificate ("kenaflow Licensing Root") from the "Intermediate CA" store of the local machine. (Use `certlm.msc`.)
9. Remove the *kenaro* licensing certificate ("kenaro Licensing Root") from the "Root CA" store of the local machine. (Use `certlm.msc`.)

! You can remove remote event receivers manually without *kenaflow* using CSOM code.

CHAPTER

TEN

GETTING STARTED

Workflows in *kenaflow* consist of a series of files, each stored in a workflow folder. (Below this folder are no files belonging to another workflow.)

The workflow files you work with are all stored as readable PowerShell script files in text format.

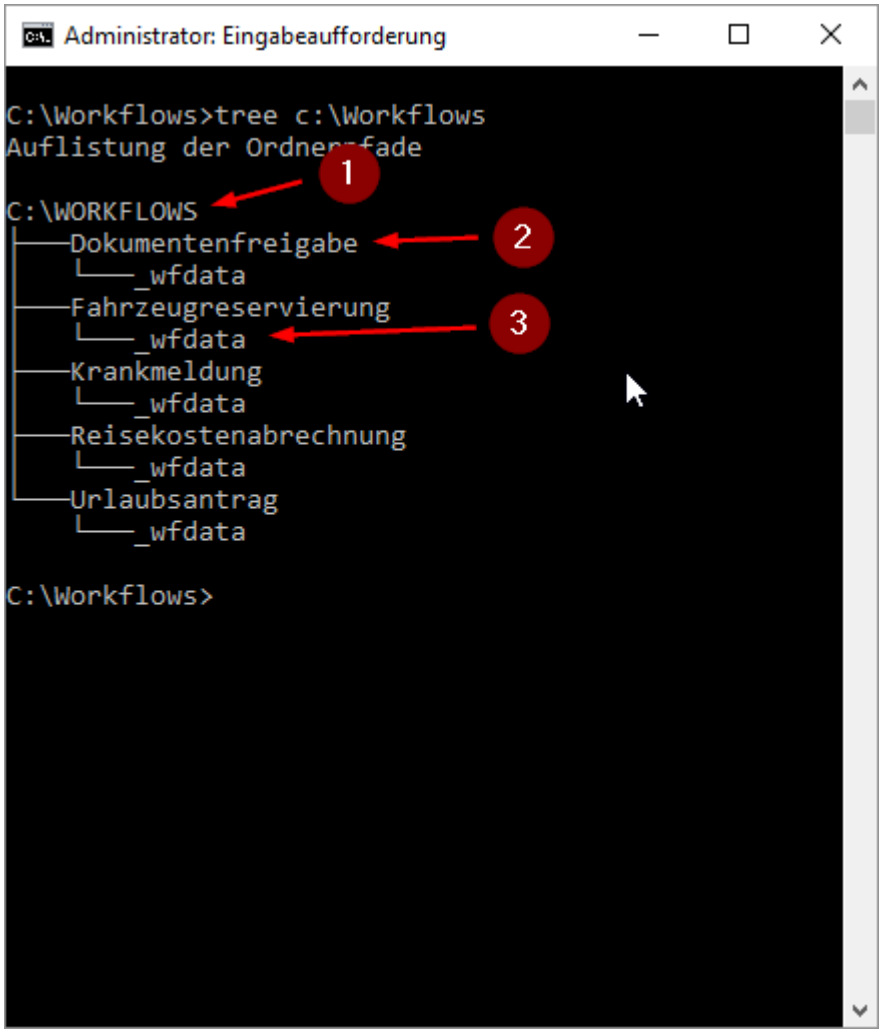
The workflow directory contains at least:

- `_wfconfig.ps1` : This is the PowerShell file that describes or defines the workflow.
- 1 or several script files that contain the actions of the workflow.

Because the files are stored as text, they can be easily managed with a version control system. We recommend GIT as the version control system for *kenaflow* workflows.

Each workflow consists, as described, of files in a workflow folder. The workflow folders are in a directory, which is here referred to as the "Workflow Root" directory. (The name of the directory on the hard disk can be chosen as desired.)

Such a folder structure can look like this:

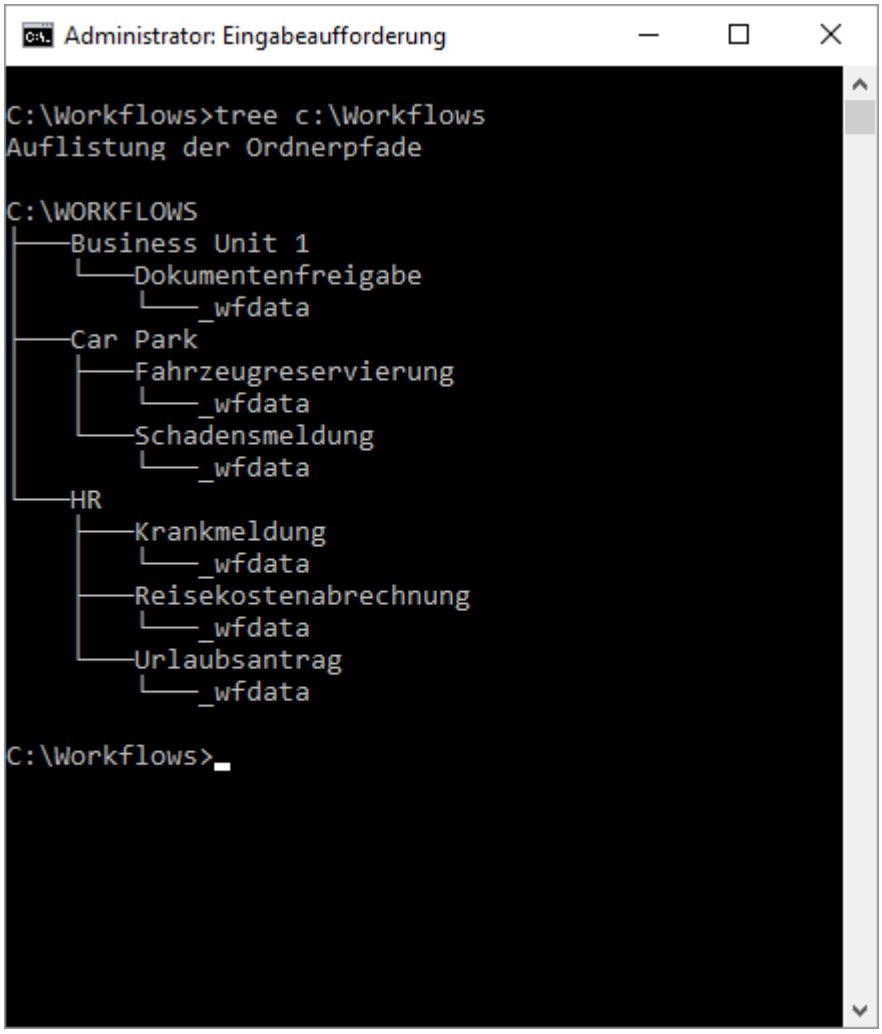


Here (1) is the workflow root directory. (2) is the directory of the "Dokumentenfreigabe" ("Document Sharing") workflow.

The directories named `_wfdata` (3) contain files written by *kenaflow* during workflow execution. **These files should not be changed directly.** By own changes *kenaflow* could not work as expected. However, a loss of this directory is not critical. It will be rebuilt if necessary.

In the example shown you can see a total of 5 workflows. You can define any number of workflows in the "Standard" edition of *kenaflow*. The only limiting factor is the performance of the computer on which you run *kenaflow*.

You can also use other directory levels, such as:



It is only important that no other workflows are stored under a workflow directory. The files in the workflow directory define the workflow.

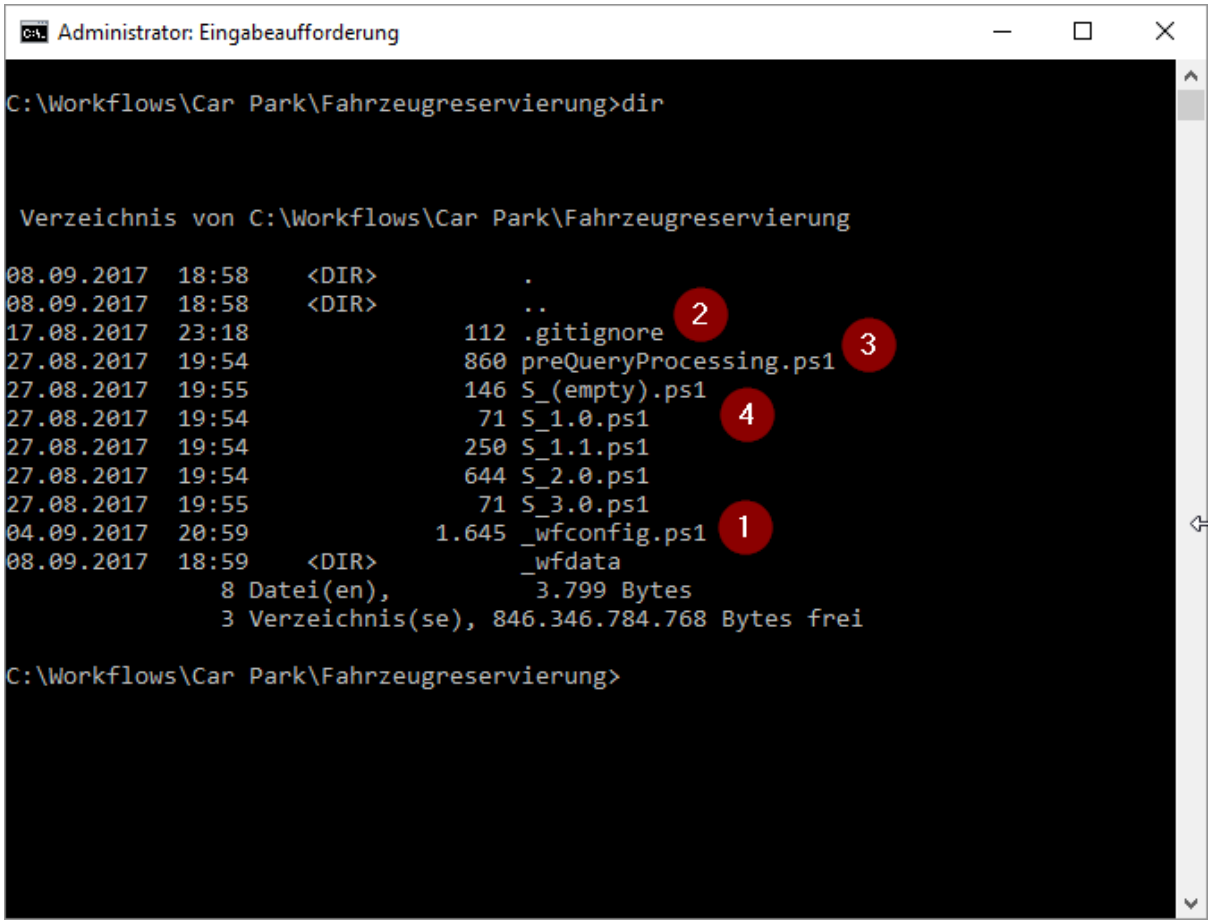
The most important file is the workflow configuration file `_wfconfig.ps1`.

This file contains the description of the workflow, e.g. the type of workflow, the name, the ID, the workflow steps (for State Machine workflow) and many other parameters.

A description of all parameters of the workflow configuration file can be found here:

The workflow directory also contains the workflow scripts that are executed by *kenaflow*.

Here is an example of the files in a workflow directory:



1. is the workflow configuration file `_wfconfig.ps1`
2. is a *kenaflow* created .gitignore-file that tells the GIT version control system to ignore the files in the sub-folder `_wfdata`.
3. is a "Pre-Query" script that executes before the list items are queried when the workflow is executed.
4. are several workflow step scripts whose file name starts with `S_` followed by the step key. More about that later.

To create a new workflow with *kenaflow* you can either start with a copy of an existing workflow or with an empty workflow.

10.1 Workflow types

You can create a new workflow using `kenaflow.exe`.

kenaflow currently has three templates for new workflows:

- List Workflow
- Site Workflow
- State Machine Workflow

10.2 Little video excursion

Coming Soon

10.3 What is a Workflow

We would like to explain what we mean by "workflows" so that it is clear that we are talking about the same thing below.

A "workflow" in our sense is a business process in the company that is technically and digitally executed, with the technical component - the "workflow engine" - taking control of the process and coordinating people and IT systems.

Every company has countless business processes. One could say that these processes are "applications" - or: algorithms - which do not run in computers, but in the company itself. If the processes are implemented digitally, the focus shifts. Now computers are particularly involved, and if this is only the case for controlling the process.

One example:

> One such process is the request for holiday. In the past, a paper based post folder had to be forwarded from the applicant to the deputy, from there to the supervisor and from there to the human resources department. At each station, the person responsible can approve or reject the request. Depending on the result (approval or rejection), the process takes a different route or comes to an end at different points. In any case, the applicant must be informed.

All large companies used to have internal postal systems. Of course, this has to be done digitally nowadays.

In addition, there are systems such as *Microsoft SharePoint*, which from our point of view is the best on the market. For this reason and because of our long experience with the introduction of SharePoint in companies, we have decided to develop a new workflow engine for *SharePoint*.

> Today the holiday request form is filled out as a digital form in the "Intranet" (*SharePoint*). However, the form is not very mobile as a data record - just like a post folder in the past, which lay on a desk and was not collected by the postman. Only a workflow that "virtually moves" the completed form to the various stations sets the business process in motion.

This "digital movement" is the "workflow" as we think it.

> The deputy shall be informed by email of the colleague's request for holiday. He/she can reply directly to the email: The reply is stored in a mailbox and read and processed by the Workflow Engine (Email Workflow). Or the deputy clicks on a link in the email that directly triggers a response. Or the deputy clicks another link and the SharePoint form opens in the browser where he/she can enter the answer. Once the answer has been entered, the request is forwarded to the superior - if approved. If the request is rejected, the requester is informed and the workflow is complete. For the superior, the process is the same as for the deputy. Finally also for the personnel department. At the end, the applicant is informed that the application has been approved. Done.

But things are getting more complicated. Here is also the "old house mail" from the race. If we imagine that today there are teams distributed all over the world, then the transport of paper from one person to another is over. And that's just the simplest reason why things have to be different today.

> All of them could be relevant in the holiday request! To query whether a customer project is still running. Or whether certain persons have already applied for leave. Or whether the employee still has enough remaining holiday.

kenaflow makes it possible to digitally implement these business processes with SharePoint stored data.

10.4 Workflow Processing

This chapter is about the processing principles behind a *kenaflow* workflow.

Such a *kenaflow* workflow is - as already told - a bunch of PowerShell scripts.

And also already told we offer workflows for different purposes, especially SharePoint, E-Mail and PowerShell.

Workflow must perform their work on a regular basis. However, not every system reports when there is work to be done.

SharePoint, for example, is able to report a change to a list item to other software. This is called an "Remote Event Receiver".

kenaflow is such a "Remote Event Receiver" for SharePoint. It registers a kind of web hook in SharePoint - *Online* as well as *2013*, *2016* and *2019.

Furthermore *kenaflow* workflows can act as webhooks: They can receive for example "unstructed data" as we call it.

Please read this article for details: *Remote Event*

Other systems like E-Mail mailbox servers do not inform a third party system about incoming mails.

Therefore *kenaflow* executes workflows on a reguar basis.

Every workflow has a setting called "TBE" = "Time Between Execution". This is the amount of time (seconds) > 0 that must elapse between two **starts** of a workflow.

A "TBE" < 0 means that a workflow is not executed on a regular basis. This is useful for SharePoint workflows that use "Remove Events".

A "TBE" of 0 means that a workflow will start as soon as possible after it is finished.

"TBE" is not guaranteed. Because ALL workflows are queued it can take much longer to be executed. Or much more time between two starts elapses. Because of the queue.

"TBE" an be combined with "CRON" to schedule workflows on a time pattern basis.

Regular execution ("TBE" >= 0) can be combined with "Remove Event" based execution.

Starting with *kenaflow* 4.0 we offer "permanent executing" workflows: There each "permanent" workflow runs in a separate process in the operating system.

All non-permanent workflows are queued for execution. If you have lots of workflows only some of them will run in parallel. You can configure the amount of parallel workflows.

A workflow can have a priority. This can be used to influence the workflow position in the queue. Normally, an executed workflow is added to the back of the queue and takes some time to run again. As already told the "TBE" - time between executions - can be much larger than executed with lots of workflows are in the queue... BUT with "priority" you can control it's position. Based on it's "TBE" and priority **relative** to the "TBE" and priority of all other queued workflows the next execution of a workflow gets re-calculated / re-scheduled permanently.

Details: *Workflow Priority*

For more flexibility it is possible to define several queues for workflows. We call it "slots". The workflows that are assigned to a "slot" share the same queue.

Per slot the amount of parallel workflows can be configured.

Details: *Execution Slots*

10.5 Workflow Types

The following sections introduce the types of workflows that can be created and run by *kenaflow*.

They are:

1. **State Machine Workflows**:

   Workflows of this type are distinguished by the fact that SharePoint list items have a state that is managed by the workflow and tells the workflow what "to do next" with the list item.

1. **List Workflows**:

   List workflows have only one workflow script that will be applied to all SharePoint list items queried by the workflow, e.g. SharePoint list items that have changed since the last run of the workflow can be selected.

1. **Site Workflows**:

   Site workflows are a type of workflow that is independent of a list item.

1. **Email Workflows**:

   An e-mail workflow processes the e-mails in a mailbox and executes the workflow script for each e-mail found.

1. **PowerShell Workflows**:

   PowerShell workflows are regularly executed PowerShell scripts, but their special feature is that they can also be executed via remote event. This means that data is sent to the workflow to be executed.

10.5.1 State Machine Workflows

10.5.1.1 Example

In your company, you want to use *kenaflow* to implement a hiring workflow that coordinates the tasks to be performed before and after hiring an employee so that the employee can do his or her job. Order a notebook, reserve a workstation. Print business cards. Order an Office 365 account. Each of these tasks is a "state" in which the process is located. Each state is processed and confirmed by the responsible person. Then the next state is executed, i.e. the responsible person is informed.

10.5.1.2 Getting Started Knowledge

Workflows of this type are distinguished by the fact that the SharePoint list items are assigned to a state that is managed by the workflow and tells the workflow what state the SharePoint list item is in.

In the context of the state machine workflow actions are executed. If necessary, the state can be changed by external actions, so that the execution of the workflow is influenced.

A "State Machine" workflow is connected to exactly **one** SharePoint list.

The workflow state is stored as lookup column in the SharePoint list item. The workflow checks the state information and then decides which workflow state should be executed. This allows complex workflows to be constructed by giving the SharePoint list item from one workflow state to the next.

Each workflow state owns a workflow script. This contains the actions that belong to the state. The workflow script of a state can set the SharePoint list item to a different workflow state.

The workflow script belonging to the workflow state is stored as individual files in the workflow directory. The filenames start with `S_` - and end with `.ps1`, because they are PowerShell scripts.

The workflow states that belong to a specific workflow, are defined in the workflow configuration file `_wfconfig.ps1`.

For each workflow state, it is possible to define which queries (see section *Queries*) must correspond to a list item in order to be processed by the workflow. It is also possible for each workflow state to process all items that are in this state. Or only execute the workflow script on items that are changed since the last runtime of the workflow state.

Any list item found by querying a workflow state is processed by the state workflow script.

State Machine workflows can have any number of workflow states.

Various objects are passed to the workflow (PowerShell) script:

- the workflow configuration
- the list of workflow states
- the current SharePoint Web
- the current list item
- a list of the workflow configuration data stored in SharePoint

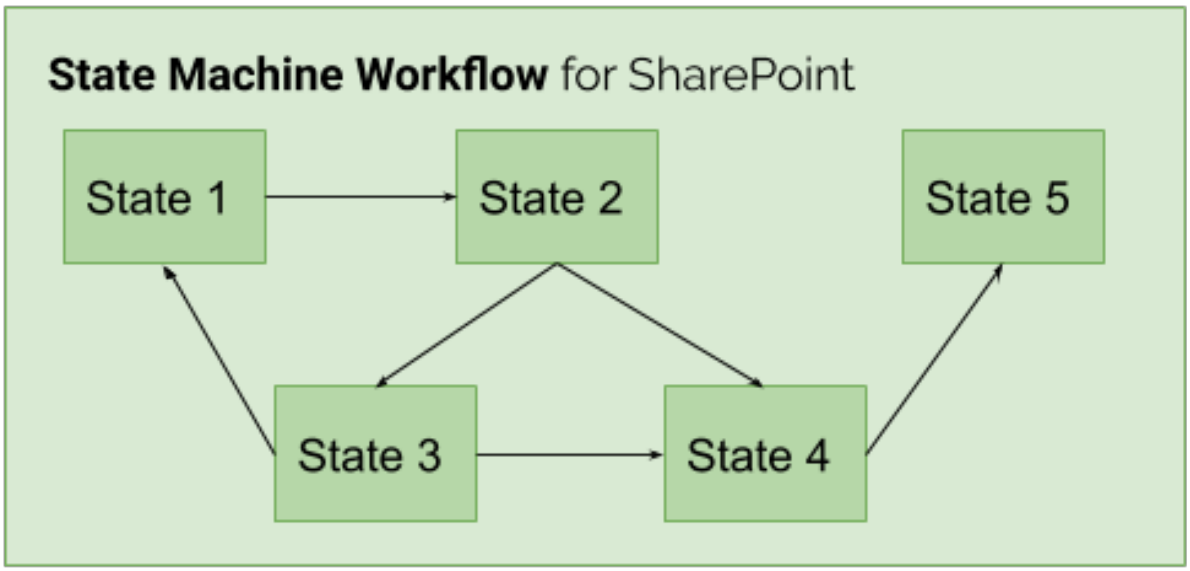
Each state machine workflow can have a "pre-query" script executed by *kenaflow* before the list items are queried and the workflow scripts are executed.

Each List-workflow and Site-workflow consists of exactly one workflow state, which is executed each time the workflow is executed.

In the case of Site-workflows, the one workflow state is executed exactly **once** per workflow execution.

In the case of List-workflows, the workflow state is executed for **each list item** that is determined by the workflow query.

10.5.1.3 Background Knowledge



The "State Machine" workflow type has some special features since the workflow execution is distributed among several scripts.

The individual states of the workflow are executed one after the other. For this purpose, the list items to be processed are determined separately for each state.

There are 3 ways to query list items to be processed. For each workflow state this can be choosen individually.

- by "LASTMODIFIED" : queries all list items that were modified since the last workflow run
- by "QUERY" : using *kenaflow* specific *Queries* or using CAML
- by "ALL" : all list items are processed

The order of the state can be controlled via the workflow configuration.

If the "LASTMODIFIED" type is used for a state, list items are processed that have been changed since the last execution of this workflow state.

The order of workflow states can be set via `_wfconfig.ps1`. Each state has the optional field `ExecutionOrder`. If this is not filled, 0 is assumed. Except for the state with the state code (empty): for this `ExecutionOrder = -1` is assumed so that this state is always executed first in the standard.

All states with the same `ExecutionOrder` are sorted by their state code.

Since each workflow can be limited in its runtime, it is possible that a workflow will be terminated before all states have been completed.

kenaflow remembers the last state performed and repeats the sorting of states so that the next time the last executed state is started.

The execution time for each workflow state can be limited.

- When the time is up, *kenaflow* remembers the last list item ID of the last processed item to continue with the next larger ID on the next run.
- It is also possible to instruct *kenaflow* to remember the modification time of the last item to start at this point in time at the next run.

After executing a state, *kenaflow* remembers the date and time of the last execution. This point in time is relevant for the determination of the list items to be processed according to the principle LASTMODIFIED.

!!!! For workflows that are aborted by *kenaflow* - e.g. due to exceeding the maximum execution time -, the last processing time is stored. Thus LASTMODIFIED may not deliver the desired result in the next execution!

Workflows that are terminated because the *kenaflow* application closed - for example, by restarting the computer,- will usually not save the last execution time. This is not possible because the operating system terminates the software. *kenaflow* tries to save the time. However, there is no guarantee.

LASTMODIFIED should generally be omitted as the query type for workflow states! Rather, smart queries should be developed for each state, based on list items properties to determine whether or not items need to be processed.

For each workflow state, you can specify how many list items should be retrieved per batch. This setting is either inherited from the workflow, if it exists or from the global configuration of *kenaflow*. By default, it is 1000. It is important to ensure that this value does not collide with SharePoint's "List View Threshold". This can lead to errors. The SharePoint standard "List View Threshold" is 5000.

Users and processes like workflows are constantly working in SharePoint. List items change.

kenaflow uses queries to determine the list items to be processed. If the query has taken place, it is not executed again until all determined elements have been processed. If the elements change in between, they will still be included in the query being processed.

With the "ItemReCheckTime" setting, you can set for each workflow state the point in time (in seconds from the start of the workflow state) at which each element is to be checked again to see whether processing is still necessary.

The setting is inherited from the workflow if it is missing or if it has a value of -1, if it exists there, or from the global configuration of *kenaflow*.

A value of 0 indicates that each list item will be checked without delay, whether it still needs to be processed.

10.5.2 List Workflows

10.5.2.1 Example

You have a SharePoint document library where users are asked to fill some taxonomy fields to classify documents. You cannot make the fields "required" because this is a problem when you upload multiple documents at once.

With *kenaflow* you create a workflow that checks every file every night and send the last editor an email that asks them to fill in the document classifications.

10.5.2.2 Getting Started Knowledge

Workflows of this type have only one workflow script that will be applied to all SharePoint list items queried by the workflow.

E.g. SharePoint list items that have changed since the last run of the workflow, can be queried. In this case, the workflow script will be applied to all these modified SharePoint list items.

There is no distinction at workflow level between the status of the SharePoint list items.

A "List Workflow" is connected with exactly **one** SharePoint list.

"List Workflows" have exactly **one** workflow script that is applied to the SharePoint list items determined by a condition or query.

For every list item, that applies to the query or condition of the list workflow, a workflow script will be executed.

The script is given various objects:

- the workflow configuration
- the current SharePoint Web
- the current list item
- a list of the workflow configuration data stored in SharePoint

Each list workflow can have a "pre-query" script run by *kenaflow* before the list items are queried and the workflow scripts are executed.

There are 3 ways to query list items to be processed:

- by "LASTMODIFIED" : queries all list items that were modified since the last workflow run
- by "QUERY" : using *kenaflow* specific *Queries* or using CAML
- by "ALL" : all list items are processed

10.5.2.3 Background Knowledge

The execution time for a workflow can be limited.

- When the time is up, *kenaflow* remembers the last list item ID of the last processed item to continue with the next larger ID on the next run.
- It is also possible to instruct *kenaflow* to remember the modification time of the last item to start at this point in time at the next run.

After executing a list worklfow, *kenaflow* remembers the date and time of the last execution. This point in time is relevant for the determination of the list items to be processed according to the principle LASTMODIFIED.

!!!! For workflows that are aborted by *kenaflow* - e.g. due to exceeding the maximum execution time -, the last processing time is stored. Thus LASTMODIFIED may not deliver the desired result in the next execution!

Workflows that are terminated because the *kenaflow* application closed - for example, by restarting the computer,- will usually not save the last execution time. This is not possible because the operating system terminates the software. *kenaflow* tries to save the time. However, there is no guarantee.

LASTMODIFIED should generally be omitted as the query type for list workflows! Rather, smart queries should be developed, based on list items properties to determine whether or not items need to be processed.

For a list workflow, you can specify how many list items should be retrieved per batch. By default this setting is either inherited from the global configuration of *kenaflow*. By default, it is 1000. It is important to ensure that this value does not collide with SharePoint's "List View Threshold". This can lead to errors. The SharePoint standard "List View Threshold" is 5000.

Users and processes like workflows are constantly working in SharePoint. List items change.

kenaflow uses queries to determine the list items to be processed. If the query has taken place, it is not executed again until all determined elements have been processed. If the elements change in between, they will still be included in the query being processed.

With the "ItemReCheckTime" setting, you can set the point in time (in seconds from the start of the workflow) at which each element is to be checked again to see whether processing is still necessary.

The setting is inherited from the workflow if it is missing or if it has a value of -1, if it exists there, or from the global configuration of *kenaflow*.

A value of 0 indicates that each list item will be checked without delay, whether it still needs to be processed.

10.5.3 Site Workflows

10.5.3.1 Example

In your company, you enable employees to share SharePoint data with other people in the company. This also means that nobody knows exactly which person has access where. - On a project page, you therefore implement a *kenaflow* site workflow that resets permission inheritence every night on all SharePoint lists on the Web each list item with individual permissions.

10.5.3.2 Getting Started Knowledge

Workflows of this type are executed without reference to a specific SharePoint list but with a connection to a SharePoint web site.

This also makes it impossible to apply the workflow script to individual list items. Rather, the "Site" workflow script is executed regularly - according to the workflow configuration.

Because the site workflow is not directly linked to a SharePoint list and its items, the workflow developer must realize the access to SharePoint data itself.

The site workflow is ideal for batch processing at some point in time, here's an example.

During workflow execution, the site workflow script is called with the following parameters:

- the workflow configuration
- the current SharePoint Web
- a list of the workflow configuration data stored in SharePoint

10.5.4 Email Workflows

10.5.4.1 Example

You want to realize a ticket system with *kenaflow*. Therefore incoming mails must be processed. An automatic answer needs to be send to the ticket author. A unique ticket ID is given.

You cannot realize this scenario with SharePoint "incoming mail" features. - With *kenaflow* it is easy.

10.5.4.2 Getting Started Knowledge

An e-mail workflow processes the e-mails in a mailbox and executes the workflow script for each e-mail found.

You can connect a *kenaflow* email workflow to a mailbox hostet in Exchange (2007, 2010, 2013, Online) or accessable with IMAP or POP3.

kenaflow will mark processed mails as "read" and only process "unread" mails. - In case of POP3 this is not possible because mails cannot be marked as "read". They have to be removed in order to prevent multiple executions. So you should not use POP3.

The email is processed by a workflow script and is given as object with these properties:

```
public class kenaflowMailMessage
{
        public string Id { get; set; }
        public string TextBody { get; set; }
        public string HtmlBody { get; set; }
        public string Subject { get; set; }
        public IEnumerable<kenaflowMailAttachment> Attachments { get; set; }
        public MailAddress Sender { get; set; }
        public List<MailAddress> CcRecipients { get; set; }
        public List<MailAddress> Recipients { get; set; }
        public List<KeyValuePair<string, string>> Headers { get; set; }
}
```

Attachments are of this type:

```
public class kenaflowMailAttachment
{
        public MemoryStream Content { get; set; }
        public string FileName { get; set; }
}
```

10.5.5 PowerShell Workflows

10.5.5.1 Example

Your companies event management system can send changes of events, dates and times as CSV string to a web service. That's all. - But you need to update you SharePoint based calendar! - No problem with *kenaflow*: You can use a PowerShell workflow to have an http(s) endpoint for any data. This data is received by *kenaflow* and will be executed with the workflow script.

10.5.5.2 Getting Started Knowledge

During execution the workflow script gets the following objects

- Workflow configuration
- Config values (from `_wfconfig.ps1`)
- Event data : data send using http endpoint of the workflow - if activated

You can use the PowerShell workflow to run the workflow script regularly. This makes it possible to perform recurring tasks.

PowerShell workflows are not connected to SharePoint or an e-mail mailbox.

PowerShell workflows can get data via http endpoints.

10.6 Create workflows

To start with a new workflow we recommend to use the templates contained in `kenaflow.exe`.

How to create a new workflow:

- Go to the workflow root directory and create a new subdirectory.
- Change to this directory.
- Execute `kenaflow.exe`:

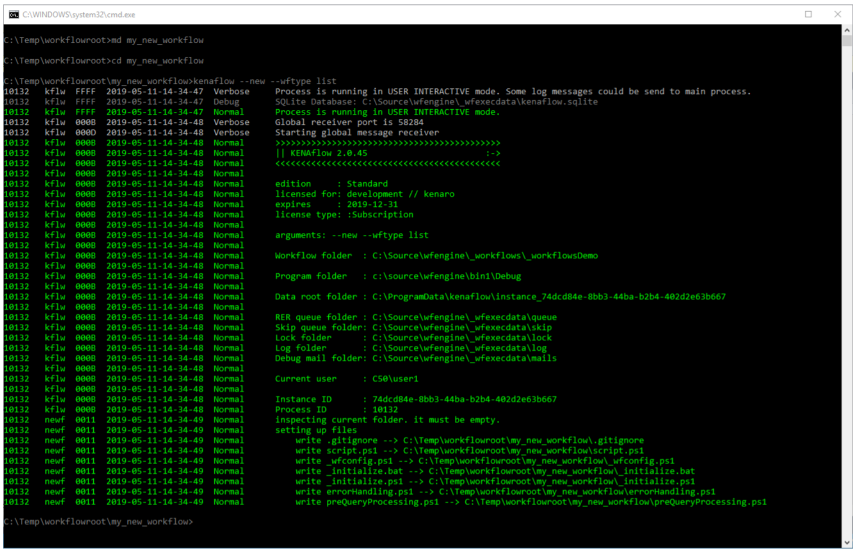
```
kenaflow.exe --new --wftype <workflow-type>
```

<workflow-type> can be one of the following values:

- list
- statemachine
- site
- email
- posh

Example:

```
kenaflow.exe --new --wftype list
```



- This command will fill the current directory.
- You must edit the workflow configuration file `_wfconfig.ps1` to make some settings.
- You can now open the workflow script file in the PowerShell ISE or in Visual Studio Code.
- You must set the SharePoint address ("Web") and the list on which the workflow operates.

Each new workflow is initially disabled (`Enabled = $false`) so *kenaflow* would not execute it. This must be changed manually.

10.7 Structure of a Workflow

There are some basic rules that workflows have to follow.

1. "Workflows" are folders inside the workflow root folder.
2. The main configuration file for each workflow is `_wfconfig.ps1`. This name cannot be changed.
3. Some data is stored in the sub folder `_wfdata`. Ths name cannot be changed. It is not important to keep this folder. Everything in it can get lost. It contains "last run" timestamps and a serialized version of `_wfconfig.ps1`.
4. The workflow folder contains a workflow script. It's name can be configured in the `_wfconfig.ps1`.
5. In case of "state machine workflows" there can be one or more workflow state scripts. All of them start with "S_", but the name can be configured in `_wfconfig.ps1`.
6. For SharePoint workflows you can have a "pre query" script in the workflow folder. It's usage and name is configured in `_wfconfig.ps1`.
7. For all workflows except "PowerShell" workflows you can have a separate "error handling" script that is executed in case of unhandled errors. It's usage an name is configured in `_wfconfig.ps1`.
8. When using the kenaflow embedded workflow templates you will have these files additionally. They can be removed.

- `.gitignore`
- `_initialize.bat` : This is a shortcut to the next file...
- `_initialize.ps1` : This can be used to create workflow lists, the workflow state field (in case of "state machine" workflows) and the *kenaflow* defined permission sets.

1. You can have other files in the workflow folder - as you want.

10.8 Workflow Scripts

The following information applies to each of the script files for

- Site Workflows
- List Workflows
- State Machine Workflows
- Email Workflows
- PowerShell Workflows

Workflow scripts do the actual work in a workflow. These scripts have to be created by the workflow developer.

All workflow types except "State Machine Workflow" have 1 workflow script.

State machine Workflows have a workflow script per workflow state.

Every workflow script can execute all PowerShell commands. This includes all the cool & fancy .NET possibilities that are reachable by PowerShell code.

This is one of *kenaflow's* strengths: you are not tied to the possibilities that we provide you.

However, *kenaflow* offers **a number of cmdlets** for PowerShell, which are described in the chapter *Script Reference*.

Workflow scripts always start with a param command used by the *kenaflow* runtime environment to pass necessary objects to the workflow script. These lines are specific to the workflow type.

The next line we call "magic line" because it enables debugging of workflow script from within the workflow execution. You could remove this line but in this case you will not be able to debug workflow scripts with the appropriate *kenaflow* runtime context!

10.8.1 Magic Line

```
    if($wf-eq$null){Import-Module "<<directory>>\kenaflow.runtime.dll";Invoke-
↪Kenaflow;exit}
```

`<<directory>>` is the path to the assembly `kenaflow.runtime.dll`.

The cmdlet `Invoke-Kenaflow` does the actual magic for Debugging. You find a description here: *`Invoke-kenaflow`*.

10.8.2 Other Workflow Scripts

There are two more workflow scripts that need to be named:

10.8.2.1 Pre-Query Script

In case of *State Machine* and *List* SharePoint workflows it is possible to specify a script **before** executing the SharePoint list item queries for the actual workflow execution.

It is configured in the worfklow configuration file `_wfconfig.ps1`

If it is not configured than this is skipped.

IF the workflow script is executed for an remote event the pre-query script is *not* executed!

What you do in these scripts is the workflow developers thing. E.g. you could read or write data from or to files in the workflow folder. Or you stop the actual workflow execution by some conditions using *`Stop-KFScript`*.

The default pre-query script is named `preQueryProcessing.ps1` and has the following default content:

```
    param($wf, $config, $eventData)

    # This is the code you need to debug. It does not hurt the workflow during␣
↪prdoction but can be removed of course.
    if($wf-eq$null){Import-Module "<<directory>>\kenaflow.runtime.dll";Invoke-
↪Kenaflow;exit}
```

Here is the description of the script parameters:

- `$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.
- `$config` is a PowerShell hashtable that contains the additional configuration stored in SharePoint. See chapter *Workflow Lists*.
- `$eventData` is the object that contains the data of the remote event. If the script is executed outside of a remote event this parameter is `$null`.

10.8.2.2 Error Handling Script

All workflow types can specify an error handling script.

It is configured in the worfklow configuration file `_wfconfig.ps1`

If it is not configured than the following mechanism is not present.

The error handling script is executed in case of uncaught exceptions. If you execute a workflow script this could lead to errors in PowerShell - as in every other programming language or application. Especially during development.

If an error / exception occurs the script is aborted. Normally *kenaflow* handles the error. The default handling is to stop the workflow and inform the developers - if configured - via email.

If configured the error handling script is used to process the exception.

The default error handling script is named `errorHandling.ps1` and has the following default content. The first line for the parameters differes for the different workfow types. Here is the line for **SharePoint list workflows** and **SharePoint state machine workflows**.

```
    param($wf, $config, $exception, $list, $item, $eventData)

    if($null-eq$wf){write-error "You need to run a workflow script and force an␣
↪error.";exit;}


    #Return one of these values:
    # 0 or "ThrowException" or [KenaflowErrorHandlingResults]::ThrowException
    # 1 or "Okay" or [KenaflowErrorHandlingResults]::Okay
    # 2 or "StopItemWithSuccess" or␣
↪[KenaflowErrorHandlingResults]::StopItemWithSuccess
    # 3 or "StopStateWithSuccess" or␣
↪[KenaflowErrorHandlingResults]::StopStateWithSuccess
    # 4 or "StopListWithSuccess" or␣
↪[KenaflowErrorHandlingResults]::StopListWithSuccess
    # 5 or "StopWorkflowWithSuccess" or␣
↪[KenaflowErrorHandlingResults]::StopWorkflowWithSuccess
    # 6 or "StopItemWithFailure" or␣
↪[KenaflowErrorHandlingResults]::StopItemWithFailure
    # 7 or "StopStateWithFailure" or␣
↪[KenaflowErrorHandlingResults]::StopStateWithFailure
    # 8 or "StopListWithFailure" or␣
↪[KenaflowErrorHandlingResults]::StopListWithFailure
    # 9 or "StopWorkflowWithFailure" or␣
↪[KenaflowErrorHandlingResults]::StopWorkflowWithFailure
    #
    # "ThrowException" is default in case nothing is send back
    #
    # "Okay" indicated that the error handling script has handled the exception␣
↪and the engine must not take care.
    #
    # You cannot send other data back to _kenaflow_!

    return 0
```

Here is the description of the script parameters:

- `$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.
- `$config` is a PowerShell hashtable that contains the additional configuration stored in SharePoint. See chapter *Workflow Lists*.
- `$exception` contains the exception that caused the abortion of the workflow script.

- `$list` contains the current SharePoint list (for **State Machine Workflows** or **List Workflows**. IT DOES ONLY EXIST for workflow scripts of the named workflows types. It does not exist for pre-query scripts.
- `$item` contains the current SharePoint List item (for **State Machine Workflows** or **List Workflows** or **Email Workflows**). It is `$null` in case of executing exceptions from the pre-query script (for **State Machine Workflows** or **List Workflows**) and it DOES NOT EXIST AT ALL for **SharePoint Site Workflows** and **PowerShell Workflows**
- `$eventData` contains the current remote event data. Or it is `$null` if there is no event data. For **Email Workflows** this parameter does not exist!

For **SharePoint Site Workflows** and **PowerShell Workflows** the first line looks like this:

```
param($wf, $config, $exception, $eventData)
```

For **Email Workflows** the first line looks like this:

```
param($wf, $config, $exception, $item)
```

!! If an error occured inside the error handling script an exception is passed to the workflow engine. There is no "error handling script for the error handling script". (Yes, we got this question once!)

In case of an *Email* workflow or an SharePoint *List* workflow `StopStateWithSuccess` behaves as `StopWorkflowWithSuccess` and `StopStateWithFailure` behaves as `StopWorkflowWithFailure`.

In case of an *PowerShell* workflow `StopItemWithSuccess` and `StopStateWithSuccess` behaves as `StopWorkflowWithSuccess` and `StopItemWithFailure` and `StopStateWithFailure` behaves as `StopWorkflowWithFailure`.

In case of errors inside the pre-query script `StopItemWithSuccess` and `StopStateWithSuccess` behaves as `StopWorkflowWithSuccess` and `StopItemWithFailure` and `StopStateWithFailure` behaves as `StopWorkflowWithFailure`.

`StopListWithSuccess` behaves as `StopWorkflowWithSuccess` for all scripts **other than** a SharePoint *list* workflow script or *state machine* workflow script with *multiple lists*. The same for `StopListWithFailure`: it behaves as `StopWorkflowWithFailure`.

10.8.2.2.1 Failure / Success

In case of Email workflows a "Failure" means that the email item is not marked as "read" (IMAP or EXCHANGE) or deleted (POP3).

For all workflow types "Failure" means, that the next run of the workflow is not controlled by "time between execution". The workflow should run again as soon as possible.

10.8.3 Workflow Script Parameters

Here are the parameters / objects passed to workflow scripts by *kenaflow*.

10.8.3.1 State Machine Workflow

The following objects are passed in a **State Machine Workflow** script:

```
param($wf, $states, $web, $list, $item, $config, $eventData)
```

`$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.

`$states` is an object containing all status configurations of the workflow.

`$web` is the web object on which the workflow operates.

`$item` is the current list item that the workflow is currently working on.

`$config` is a PowerShell hashtable that contains the additional configuration stored in SharePoint. See chapter *Workflow Lists*.

`$eventData` may contain the 'Remove Event' sent by SharePoint or third party systems to *kenaflow*. If the workflow script is executed outside the remote event handler this object is `$null`.

10.8.3.2 List Workflow

The following objects are passed in a **List Workflow** script:

```
param($wf, $web, $list, $item, $config, $eventData)
```

`$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.

`$web` is the web object on which the workflow operates.

`$item` is the current list item that the workflow is currently working on.

`$config` is a PowerShell hashtable that contains the additional configuration stored in SharePoint. See chapter "Workflow-specific SharePoint lists", subchapter "Workflow Config" list.

`$eventData` may contain the 'Remove Event' sent by SharePoint or third party systems to *kenaflow*. If the workflow script is executed outside the remote event handler this object is `$null`.

10.8.3.3 Site Workflow

The following objects are passed in a **Site Workflow** script:

```
param($wf, $web, $config, $eventData)
```

`$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.

`$web` is the web object on which the workflow operates.

`$config` is a PowerShell hashtable that contains the additional configuration stored in SharePoint. See chapter "Workflow-specific SharePoint lists", subchapter "Workflow Config" list.

`$eventData` may contain the 'Remove Event' sent by SharePoint or third party systems to *kenaflow*. If the workflow script is executed outside the remote event handler this object is `$null`.

10.8.3.4 Email Workflow

The following objects are passed in a **Email Workflow** script:

```
param($wf, $web, $mail, $config)
```

`$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.

`$web` this is *not* `$null` if the email workflow is connected to a SharePoint web. This is not necessary. In case of "pure" email workflows this parameter is `$null`.

`$mail` contains the email that is currently processed.

`$config` is a PowerShell hashtable that contains the additional configuration stored in SharePoint. See chapter "Workflow-specific SharePoint lists", subchapter "Workflow Config" list. If the workflow is not connected to SharePoint this will only contain the default config values from the `_wfconfig.ps1`.

10.8.3.5 PowerShell Workflow

The following objects are passed in a **PowerShell Workflow** script:

```
param($wf, $config, $eventData)
```

`$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.

`$config` is a PowerShell hashtable that contains config values from the `_wfconfig.ps1`.

`$eventData` may contain the 'Remove Event' sent by SharePoint or third party systems to *kenaflow*. If the workflow script is executed outside the remote event handler this object is `$null`.

10.9 Copy workflows

Do not always start from scratch, but start with a copy of a matching workflow.

When copying an existing workflow, be careful **not to copy the entire** workflow directory below the workflow root, otherwise *kenaflow* will directly scour the copied folder, recognize the workflow configuration file, and read it. In this case, the IDs of the original and the copied workflow will be identical. This results in only one of the workflows being executed - maybe the copied one.

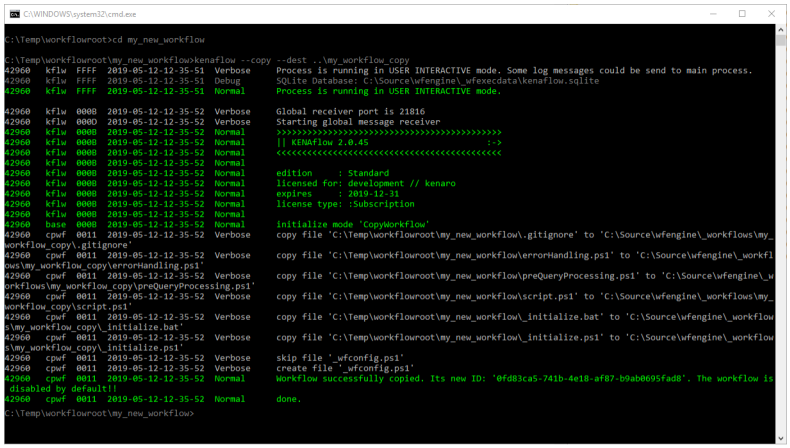
If necessary, copy the workflow to another directory **outside the workflow root**, configure the necessary attributes there and then copy the workflow back to the workflow root directory.

We have prepared a program mode in *kenaflow* to support you in copying a workflow. This will automatically set the new workflow to be disabled (`Enabled = $false`) and sets a new workflow ID.

10.9.1 Example

Here we copy the workflow in folder `c:\temp\workflowroot\my_new_workflow` to folder `c:\temp\workflowroot\my_workflow_copy` using this command:

```
kenaflow.exe --copy --dest ..\my_workflow_copy
```


C:\WINDOWS\system32\cmd.exe
C:\Temp\workflowroot>cd my_new_workflow
C:\Temp\workflowroot\my_new_workflow>kenaflow --copy --dest ..\my_workflow_copy
42960 kflw FFFF 2019-05-12-12-35-51 Verbose Process is running in USER INTERACTIVE mode. Some log messages could be send to main process.
42960 kflw FFFF 2019-05-12-12-35-51 Debug SQLite Database: C:\Source\wfengine_wfexecdata\kenaflow.sqlite
42960 kflw FFFF 2019-05-12-12-35-51 Normal Process is running in USER INTERACTIVE mode.
42960 kflw 0008 2019-05-12-12-35-52 Verbose Global receiver port is 21816
42960 kflw 000D 2019-05-12-12-35-52 Verbose Starting global message receiver
42960 kflw 0008 2019-05-12-12-35-52 Normal >>
42960 kflw 0008 2019-05-12-12-35-52 Normal || KENAflow 2.0.45 :->
42960 kflw 0008 2019-05-12-12-35-52 Normal <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
42960 kflw 0008 2019-05-12-12-35-52 Normal
42960 kflw 0008 2019-05-12-12-35-52 Normal edition : Standard
42960 kflw 0008 2019-05-12-12-35-52 Normal licensed for: development // kenaro
42960 kflw 0008 2019-05-12-12-35-52 Normal expires : 2019-12-31
42960 kflw 0008 2019-05-12-12-35-52 Normal license type: !Subscription
42960 kflw 0008 2019-05-12-12-35-52 Normal
42960 base 0008 2019-05-12-12-35-52 Normal initialize mode 'CopyWorkflow'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose copy file 'C:\Temp\workflowroot\my_new_workflow\.gitignore' to 'C:\Source\wfengine_workflows\my_workflow_copy\.gitignore'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose copy file 'C:\Temp\workflowroot\my_new_workflow\errorHandling.ps1' to 'C:\Source\wfengine_workflows\my_workflow_copy\errorHandling.ps1'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose copy file 'C:\Temp\workflowroot\my_new_workflow\preQueryProcessing.ps1' to 'C:\Source\wfengine_workflows\my_workflow_copy\preQueryProcessing.ps1'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose copy file 'C:\Temp\workflowroot\my_new_workflow\script.ps1' to 'C:\Source\wfengine_workflows\my_workflow_copy\script.ps1'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose copy file 'C:\Temp\workflowroot\my_new_workflow_initialize.bat' to 'C:\Source\wfengine_workflows\my_workflow_copy_initialize.bat'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose copy file 'C:\Temp\workflowroot\my_new_workflow_initialize.ps1' to 'C:\Source\wfengine_workflows\my_workflow_copy_initialize.ps1'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose skip file '_wfconfig.ps1'
42960 cpwf 0011 2019-05-12-12-35-52 Verbose create file '_wfconfig.ps1'
42960 cpwf 0011 2019-05-12-12-35-52 Normal Workflow successfully copied. Its new ID: '0fd83ca5-741b-4e18-af87-b9ab0695fad8'. The workflow is disabled by default!!
42960 cpwf 0011 2019-05-12-12-35-52 Normal done.
C:\Temp\workflowroot\my_new_workflow>

CHAPTER

ELEVEN

BASICS

Information on basic functions of *kenaflow* can be found on the subpages of this section.

- *Templates*

  Information about the *kenaflow* template engine for placeholder replacement.

- *Queries*

  kenaflow own query language as replacement for CAML for use in SharePoint *list* and *state machine* workflows.

- *Debugging*

  Information about workflow debugging.

- *Libraries*

  Using PowerShell libraries in *kenaflow* workflows.

- *Workflow Lists*

  What are workflow lists and what types are there.

- *Locking*

  Locking of SharePoint list items and custom locks for inter-workflow synchronization.

- *Links*

  Generating “rich links” to enable users to communicate easily with workflows by links (URLs).

- *Workflow Safety*

  kenaflow methods to prevent ever-running workflows.

- *Version Upgrade*

  Information about necessary steps for *kenaflow* version upgrades.

- *Permissions*

  Information about SharePoint permission handling.

- *Remote Event*

  Information about the various types of remote events.

- *Error Handling*

  Describs error handling methods for *kenaflow* workflows.

- *Persisting Data*

  Information about methods to persist data from *kenaflow* workflows.

- *Trigger*

  kenaflow workflows can trigger other workflows and communicate with them.

- *The kenaflow Field*

kenaflow stored some data in the "kenaflow field" of SharePoint list items in SharePoint *list* and *state machine* workflows.

- *Security Hardening*

  Details about what to do to harden the *kenaflow* instance.

11.1 Templates

kenaflow offers two methods for template processing.

The first one uses `Invoke-KFTemplate` to process strings with placeholders.

The second one uses `Invoke-KFFileTemplate` to process *Microsoft Word* and/or *Microsoft PowerPoint* files with placeholders.

11.1.1 String Processing (Invoke-KFTemplate)

We have implemented a template engine in *kenaflow* that runs easily through strings and replaces placeholders with concrete values.

This can be used, for example, to send e-mails or to create PDF files.

Example:

```
    Invoke-KFTemplate "{{Modified|yyyy-MM-dd}}"
```

A string can use the following placeholders:

`{{<listitem-field-name>}}`

`[[<config-key>]]`

`((<local-config-key>))`

`<< PowerShell >>`

Some placeholders (except PowerShell placeholders) may have a formatting "hint" after the given key, such as `{{DateColumn|yyyy-MM-dd}}`.

Allowed characters after the pipe are `|`:

```
> a b c d e f g h i j k l m n o p q r s t u v wq x y z A B C D E F G H I J
> K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 : , . - /
> *<Space>*
```

For SharePoint **user** columns, you can specify `|id` after the name of the field so that the user's ID is used instead of the display name. E.g. `{{Author|id}}`. In case of multi-value user fields the items are separated by `,` .

For SharePoint **lookup** columns, you can specify `|id` after the name of the field so that the item ID of the lookup field instead of the lookup value . E.g. `{{SomeLookupColumn|id}}`. In case of multi-value lookup fields the items are separated by `,` .

For SharePoint **taxonomy** columns, you can specify `|id` after the name of the field so that the item ID of the taxonomy field instead of the taxonomy value . E.g. `{{SomeTaxonomyColumn|id}}`. In case of multi-value taxonomy fields the items are separated by `,` .

For SharePoint **choice** cloumns with multiple values the items are separated by `,` .

For **date** fields, the date format can be specified after the pipe character `|`. E.g. `{{Created|yyyy-MM-dd}}`

For **bool** fields ("Yes/No fields") you can specify string values for "yes" and "no", e.g. `{{YesNoColumn|Why not!|No way!}}`

> Here you will find a list of the possible format types: https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-date-and-time-format-strings

To use "{{", "}}", "[[", "}}", "((" and "))" as text you need to escape them with "" (backslash character) befor each brace character.

The PowerShell placeholders `<< PowerShell >>` are disabled by default in the global configuration (setting `allowPowerShellPlaceholders`). If it is set to `False` there the workflow cannot overwrite it. But if it set to true it is possible to enable it for a certain workflow using setting `AllowPowerShellPlaceholders` in the `_wfconfig.ps1`. Default is `$false`. It is *not* inherited from the global configuration!

To specify "<<" or ">>" inside the PowerShell placeholder you need to escape them: "\>\>" and "\<\<". The workflow developer is responsible that the PowerShell content - that is executed as normal PowerShell - does not do any harm to the machine. It's not possible for *kenaflow* to check the "sense" of the PowerShell code. The PowerShell need to return a stringify-able value.

You have to escape any ">" or "<" character **inside** the PowerShell placeholder with "\>" and "\<".

The parameter `-LocalConfig` of the cmdlet can be used to apply local configuration values to be replaced with placeholders like `((LocalValue1))`. With `-LocalConfig` you can pass in a PowerShell HashTable object like this:

```
$LocalConfig = @{ LocalValue1 = "Hello kenaflow" }
Invoke-KFTemplate "((LocalValue1))" -LocalConfig $LocalConfig
```

If `Invoke-KFTemplate` is used in a workflow of the following type the list item placeholders are not available because there is no default item context:

- Email Workflow
- SharePoint Site Workflow
- PowerShell Workflow ... but you can specify the parameter `-Item` to pass in a SharePoint list item, e.g. if you gathered one using PowerShell. This will work in SharePoint connected workflows, like an *email workflow* with SharePoint connection or a *site workflow*. If you use pure PowerShell ablilities to connect to SharePoint - e.g. in a *PowerShell workflow* - you can use *`Set-KFContext`* to set the SharePoint for *kenaflow*.

Here is an example for a *Site workflow* script (snippet) where a list item is loaded manually and passed to `Invoke-KFTemplate`

```
$item = Get-PnPListItem -List "Some List" -Id 1
Invoke-KFTemplate -item $item -template "{{Author}}"
```

See the Invoke-KFTemplate cmdlet section for more information: *`Invoke-KFTemplate`*

11.1.2 Special Values in the Config List Object

The workflow *config list* list was already described. Its values are available with placeholders of the style `((xxx))`.

There are so special values, that are set by *kenflow*:

- `KFInVersionConflict` : Sometimes "version conflicts" occur during workflow execution if an SharePoint item was changed while the workflow is running. Normally this is related to a user action in SharePoint. In this case the "Version Conflict" handler executes the current workflow script again.
- `KFLastItemModifiedFlag` : After processing a SharePoint list item the workflow engine restores the last editor and the last modified timestamp. But the item actually was modified by the workflow. That this happend is saved in *the kenaflow Field*.

- `KFLastExecutionType` : The reason of the last workflow made change of the item, taken from the *kenaflow* field
  - `normal` : a "normal" time-based workflow run changed the item
  - `remote event` : a "SharePoint Remote Event" exection changed the item
  - `link event` : a "Link Remote Event" execution changed the item
  - `trigger event` : a "Trigger Event" execution changed the item
- `LastEditor` : In case of a SharePoint *list* or *statemachine* workflow: The user field value of "Editor" at the start of the item processing.
- `LastModified` : In case of a SharePoint *list* or *statemachine* workflow: The date field value of "Modified" at the start of the item processing.
- `KFLastRun` : (Server local) DateTime of the last workflow run **as SharePoint compatible ISO-String**
- `KFCurrentStart` : (Server local) DateTime of the current workflow start **as SharePoint compatible ISO-String**
- `KFLastRunDT` : (Server local) DateTime of the last workflow run (as DateTime)
- `KFCurrentStartDT` : (Server local) DateTime of the current workflow start (as DateTime)
- `KFLastRunUtc` : UTC DateTime of the last workflow run **as SharePoint compatible ISO-String**
- `KFCurrentStartUtc` : UTC DateTime of the current workflow start **as SharePoint compatible ISO-String**
- `KFLastRunUtcDT` : UTC DateTime of the last workflow run (as DateTime)
- `KFCurrentStartUtcDT` : UTC DateTime of the current workflow start (as DateTime)

11.1.3 Word / PowerPoint Processing

kenaflow is able to replace placeholders in Word files (.docx) and PowerPoint files (.pptx). The result is a again a Word file or PowerPoint file.

Processing Microsoft Office files is not easy. But we managed to replace tokens in almost all file parts, such as headers, footers, tables, ... If something is missing please contact us at *support kenaro.com*.

The placeholders work the same as for `Invoke-KFTemplate` (section above), but they need an additional placeholder marker right and left around the *kenaflow* token. The additional marker characters can be configured in the global configuration, section *"cmdletConfic"*. The default markers are `%{` (left) and `}%` (right.

Example: `%{{{Title}}}%` : the field name surrounded with curly braces will insert / replace with the SharePoint list items Title field value - in case you have a SharePoint *Site* or *List* workflow or you specified a list item using parameter `-Item`.

You can also use config values from the *kenaflow Config List* using placeholders like `%{[[ConfigValue]]}%` with square braces.

Or you can specify a PowerShell HashTable object like `@{SomeValue = 1; Date = [DateTime]::Now.ToString("yyyy-MM-ss")}` with paramter `-LocalConfig`. Than you use round braces to surrond the config value: `%{((SomeValue))}%`.

You can load the Word or PowerPoint template file from:

- a SharePoint location (if the workflow is SharePoint-connected)
- a System.Byte array (.NET)
- a System.IO.Stream (.NET)
- a file path (relative to the workflow folder *or* absolute) ((The *kenaflow* service account need to have read permissions on the file.))

The produces file can be retrieved from the cmdlet as:

- a local stored file (The *kenaflow* service account need to have write permissions on the file.)
- a SharePoint location (if the workflow is SharePoint-connected)
- a System.Byte array (.NET)
- a System.IO.Stream (.NET)

Here are two nerdy examples for template processing.

Workflow script:

```
    param($wf, $web, $item, $config, $eventData)

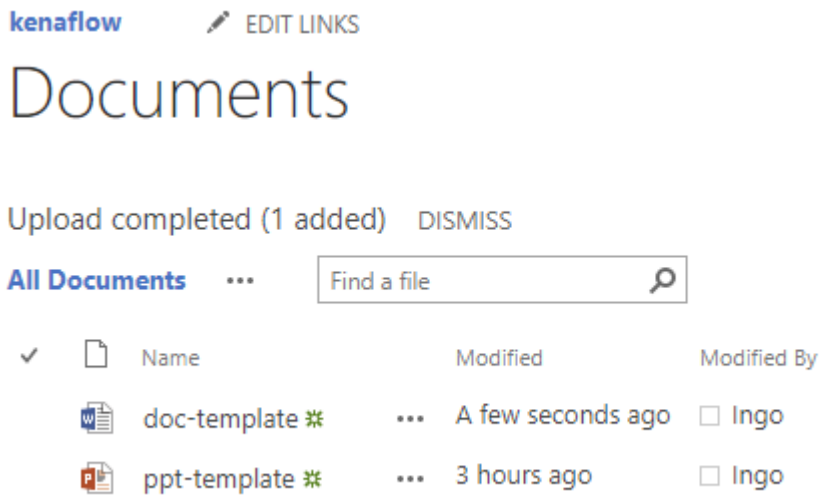
    if($wf-eq$null){import-module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Invoke-Kenaflow;exit}

    $localConfig = @{ kenaflow = "Workflow Engine" }

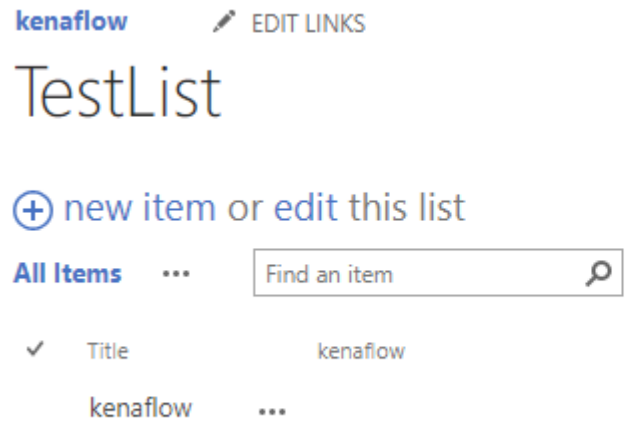
    $s = $null
    try {
        Invoke-KFFileTemplate -TemplateUrl "/sites/kenaflow/shared documents/ppt-
↪template.pptx" `
            -TemplateType PowerPoint -ToFile "$(split-path $MyInvocation.MyCommand.
↪Path)\test.pptx" `
            -LocalConfig $localConfig
        Invoke-KFFileTemplate -TemplateUrl "/sites/kenaflow/shared documents/doc-
↪template.docx" `
            -TemplateType Word -ToFile "$(split-path $MyInvocation.MyCommand.Path)\
↪test.docx" `
            -LocalConfig $localConfig
    } catch {
        Write-KFLog $_.Exception.Message
    }
```

The result files are saved to the workflow folder.

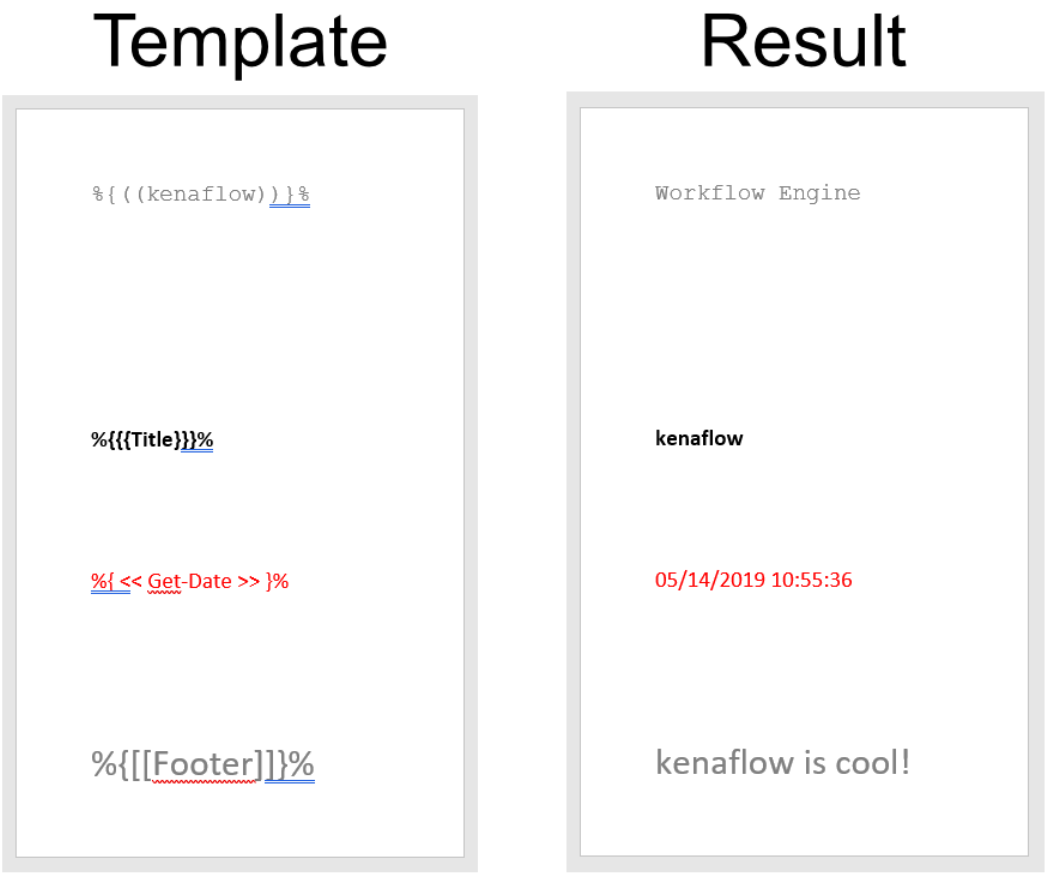
This is a screenshot of the template library:



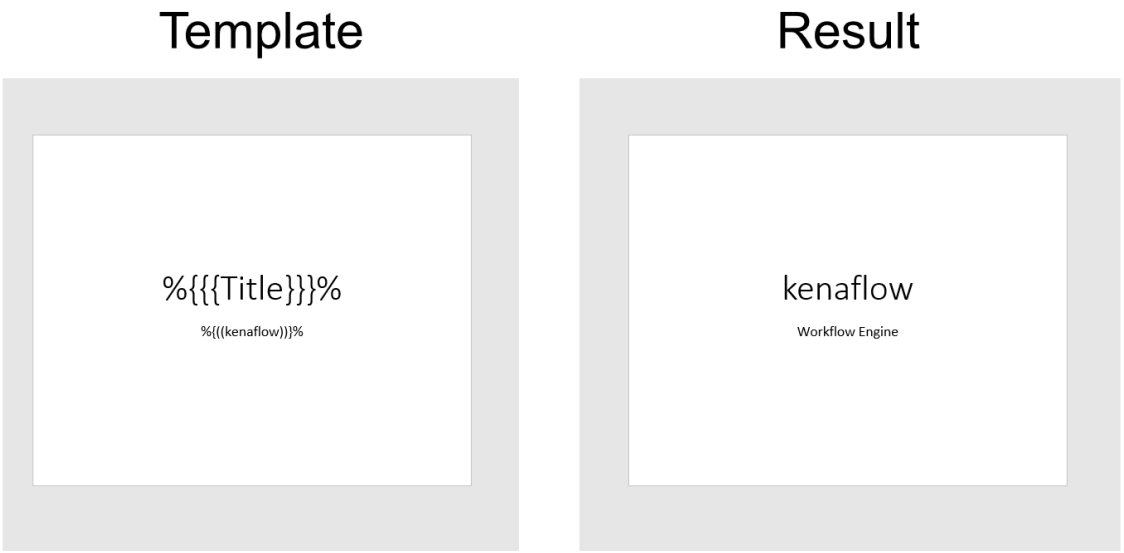
This is a screenshot of the SharePoint list to be processed:



This is the Word template with processing result:



This is the PowerPoint template with processing result:



11.2 Queries

To enable a workflow to determine which list elements are to be processed, a corresponding query of these list elements must be sent to SharePoint.

11.2.1 CAML

SharePoint experts know the SharePoint query language as "CAML" (Collaborative Application Markup Language), an XML-based structure that describes which elements SharePoint returns to *kenaflow* and then get processed by it, by calling the workflow scripts for each of these elements.

CAML is very complicated. Here you will find an introduction to the topic:

> https://msdn.microsoft.com/en-us/library/ms467521.aspx (Englisch)

You can specify CAML queries in *kenaflow*, but you must ensure that the CAML syntax is correct.

11.2.1.1 Moderation State

If you want to filter for the SharePoint "ModerationState" field than you have to be careful. The string in this field is localized. In a multi language environment you will have problems with this: `<Eq><FieldRef Name="_ModerationStatus" /><Value Type="ModStat">Approved</Value></Eq>`.

Instead use this CAML: `<Eq><FieldRef Name="_ModerationStatus" /><Value Type="ModStat">0</Value></Eq>` with these values:

- 0 = Approved
- 1 = Rejected
- 2 = Pending
- 3 = Draft
- 4 = Scheduled

11.2.2 *kenaflow* Query Language

Furthermore, *kenaflow* offers its own query language, which was kept very simple. This allows for the most common queries to be built.

Each query is composed of individual or combined sections of the following structure:

```
<sharepoint-field> <logical-operator> <data-type> <comparable>
```

For exemple:

```
{{Title}} == string "Element"
```

(This query returns all SharePoint list items whose title is "Element" exactly.)

Various such sections can be linked with the logical operators `&` for AND-operations or `|` for OR-operations.

```
{{Title}} == string "Titel 1" | {{Title}} == string "Titel 2"
```

(This query returns all SharePoint list items whose title is exactly "Title 1" OR "Title 2".)

Parentheses can and must also be used to make the query meaningful.

```
({{NumberField}} == integer 0 | {{NumberField}} == int 1) & {{Title}} !=␣
↪string 'Element'
```

(This query returns all SharePoint list items whose title is NOT "Element" (!=) and has the value 0 or the value 1 in the "NumberField" field.)

The following logical operators are possible:

| **Operator** | **Meaning** |**Description** | | — | — | — | |== | Equality operator| Both sides must be equal. | | !=| Inequality operator | Both sides must be different. | | <=| Less-or-equal to operator | The left side must be smaller than the right side. Or both sides are equal. | | >=| Larger-or-equal operator | The left side must be larger than the right side. Or both sides are equal. | | < | Less operator | The left side must be smaller than the right side. | | > | Larger operator | The left side must be larger than the right side. | | |=| Starts-with-operator | The string of the left side starts with the string of the right side. | | ~=| Contains operator | The string of the left side is contained in the string of the right side. |

!!!! There is no "Ends With" operator in SharePoint.

! When editing a "Multiple lines of text" column with the addition "Rich Text" in SharePoint, the operators ==, != and |=may reach their limits. For an alignment of strings, the operator ~= should be used. For proper use of the operators, it is recommended to use the "Text only" option or the "Single line of text" column.

On the left side, only fields of the SharePoint list, on which the workflow works, can be used. You must note that the "technical" field names are used here, which do not always match those in the SharePoint interface. If necessary, you can find out the technical name of a field by clicking on the field in the list settings and writing out the field name in the URL.

On the right side, the data type must be specified first:

| **Data type** | **Description** | | — | — | | `string`| The right side of the comparison is a string.| | `int`| The right side of the comparison is a number.| | `date`| The right side of the comparison is a date without time in the time zone of the website. The format for the following value must always be: `yyyyy-MM-dd'T'00:00:00'Z'`.| | `dateutc` | The right side of the comparison is a date without time but in the time zone UTC. The format for the following value must always be: `yyyy-MM-dd'T'00:00:00'Z'`.| | `datetime` | The right side of the comparison is a date with a time in the time zone of the website. The format for the following value must always be: `yyyyy-MM-ddTHH:mm:ssZ`.| | `datetimeutc` | The right side of the comparison is a date with a time specification in the time zone UTC. The format for the following value must always be: `yyyyy-MM-ddTHH:mm:ssZ`. | | `userid` | The right side of the comparison is the ID of a user on the SharePoint site.| | `bool` | The right side of the comparison is a boolean. Allowed for the following value are: `TRUE` or `FALSE` or `0` or `1` | | `lookupid` | The right side of the comparison is the ID of a value that appears in a lookup field.| | `lookuptext` | The right side of the comparison is the text of a lookup field. | | `modstat` | The right side of the comparison is a number 0

to 4. **THIS IS ONLY VALID FOR THE FIELD `_ModerationStatus` OF SHAREPOINT** | | `null` | For querying empty or non-empty fields. See below. |

The right side of the comparison can be one of the following options:

- a fixed value, e.g. "element title" for a string;
- a dynamic value, for example [[MyTitleField]] for a string, that comes from the workflow configuration stored in SharePoint.
- `[[Today]]` for the current date and time.
- `[[UserId]]` for the SharePoint ID of the current user. The "current User" is, however, in case of execution of a query in a workflow script of the workflow user, that is, the account in whose context *kenaflow* is currently executed.

If the data type `null` is used, only the operators == and != are allowed. No value is specified after `null`. This can be used to check whether a field is empty (==) or not (!=).

Example:

```
    {{NumberField}} == null & {{Title}} != null
```

The *kenaflow* query language is internally translated to CAML.

You can use the ConvertTo-KFCAML cmdlet to view the result of the translation.

To work with queries in *kenaflow*'s own language and with CAML, *kenaflow* provides the following cmdlets:

```
    + [`ConvertoTo-KFCAML`](#20065edc-ad51-4e22-b698-3d028fda95d3)
    + [`Get-KFNormalizedCAML`](#cf39ef76-b16d-41db-9b96-e3c4d80eacf7)
```

11.3 Debugging

There are two different debugging approaches related to *kenaflow* abilities.

The first one is debugging of workflow scripts.

The second one is debugging of remove events.

11.3.1 Debugging of Workflow Scripts

To debug a workflow script, it is either opened in Visual Studio Code or in the PowerShell ISE.

The following line must be present in the workflows script:

```
    if($wf-eq$null){Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Invoke-Kenaflow;exit}
```

A workflow script that is started outside the *kenaflow* runtime gets no parameters passed. The variable `$wf` is empty. In this case, the *kenaflow* runtime environment is initialized with the `Invoke-Kenaflow` command.

In debug mode you can use some helpful cmdlets such as

- *Invoke-KFDebugBreak*
- *Get-KFInDebug*

The workflow to which the script to be debugged belongs is executed by *kenaflow* and when the script to be debugged is reached, the debugger stops and the developer can examine the workflow.

If the script is not reached due to the workflow settings, then the debugger does not stop. This can happen because there are e.g. no SharePoint list items to match the script currently being debugged or because the query returns no result.

In case of *State Machine* workflow each script belongs to a workflow state. If you execute such a "workflow state script" the engine recognizes the assiciated state and *only executes this* state! All other workflow states are not executed *in the debugger at this time*!

In case of SharePoint *list* and *state machine* workflows you can configure multiple webs and lists that the workflow processes. If you debug such a workflows script the webs and lists are processed one after the other. You can specify the parameters `-List` and `-Web` (both or just one of them) to filter the lists and webs that are used for debugging. Both parameters accept a single string that can contain one list title and/or one web url.

11.3.1.1 Error Handling Script Debugging

To debug an error handling script of a workflow you just need to raise an exception and set a breakpoint - e.g. with *Invoke-KFDebugBreak* - in the error handling script.

To raise an error within the workflow script use:

```
throw ":-("
```

11.3.1.2 Debugging of Pre-Query Scripts

SharePoint *list* and *state machine* workflows can have a pre-query script that is executed before the items to be processed are queried from SharePoint.

You can debug these script just by adding a breakpoint to the pre-query script. You can use *Invoke-KFDebugBreak*.

11.3.1.3 Preparing a debug session

You can specify a pre-workflow execution script with `Invoke-Kenaflow`. It is executed when the environment is prepared but no workflow script was executed.

11.3.1.3.1 PowerShell Workflows

```
Invoke-Kenaflow -PreWorkflowExecution {
    param($wf, $config)

        #your script
}
```

11.3.1.3.2 SharePoint *Site* Workflows

```
Invoke-Kenaflow -PreWorkflowExecution {
    param($wf, $clientContext, $config)

        #your script
}
```

11.3.1.3.3 SharePoint *List* and *State Machine* Workflows

```
Invoke-Kenaflow -PreWorkflowExecution {
    param($wf, $clientContext, $states, $config)

        #your script
}
```

11.3.1.3.4 Email Workflow with SharePoint Connection

IMAP

```
Invoke-Kenaflow -PreWorkflowExecution {
    param($wf, $clientContext, $imapServer, $config)

        #your script
}
```

POP3

```
Invoke-Kenaflow -PreWorkflowExecution {
    param($wf, $clientContext, $pop3Server, $config)

        #your script
}
```

EXCHANGE

```
Invoke-Kenaflow -PreWorkflowExecution {
    param($wf, $clientContext, $exchangeWebService, $config)

        #your script
}
```

11.3.1.3.5 Email Workflow (without SharePoint Connection)

IMAP

```
Invoke-Kenaflow -PreWorkflowExecution {
    param($wf, $imapServer, $config)

        #your script
}
```

POP3

```
    Invoke-Kenaflow -PreWorkflowExecution {
        param($wf, $pop3Server, $config)

            #your script
    }
```

EXCHANGE

```
    Invoke-Kenaflow -PreWorkflowExecution {
        param($wf, $exchangeWebService, $config)

            #your script
    }
```

11.3.1.4 Remote Event Debugging

Remote Events are described here: *Remote Event*. Please read this article first and then come back here!

You cannot directly debug into a workflows script thats runs inside the dedicated "remote event runner" process.

You need to ***simulate*** the event context!

Therefore we have prepared the following process.

First you need to decide what type of remove event you want to debug:

- SharePoint Remote Event (as it would be send from SharePoint)
- Link Event (. . . that is created with `Get-KFLink`).
- Custom Data Event
- Unstructured Data Event

For each of the exists a "simulation" cmdlet that we describe below.

First you need to create a new PowerShell file in the folder of the workflow that is to debug with event data.

`debug-unstructured-data.ps1`:

```
    Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

    Invoke-Kenaflow -EventData {
            Invoke-KFUnstructuredDataEventSimulation -QueryString "key1=test1&
↪key2=test2" -Body "<node>some xml</node>" `
              -SimulatedUserName "sharepoint\ingo" -SimulatedUserPassword
↪"kr@ftwerk" -Authentication Basic
    }
```

This cmdlet `Invoke-KFUnstructuredDataEventSimulation` does all the magic. It creates the remote event from the given parameter values, starts the workflow and enables you to debug the workflow script.

The workflow script looks like this - very simple and basic:

`script.ps1`

```
    param($wf, $states, $web, $item, $eventData, $config)

    if($wf-eq$null){Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Invoke-Kenaflow;exit}
```

(continues on next page)

(continued from previous page)

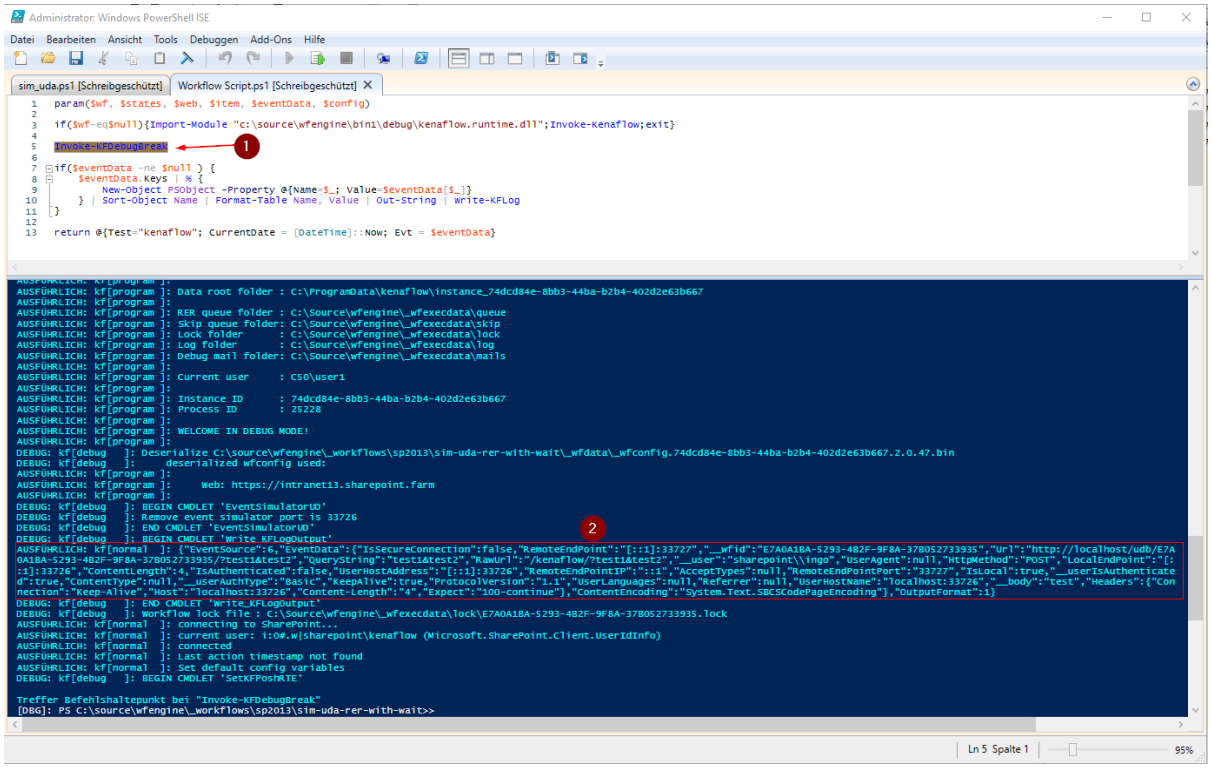
```
    Invoke-KFDebugBreak

    if( $eventData -ne $null ) {
            $eventData.Keys | % {
                    New-Object PSObject -Property @{Name=$_; Value=$eventData[$_]}
            } | Sort-Object Name | Format-Table Name, Value | Out-String | Write-
↪KFLog
    }
```

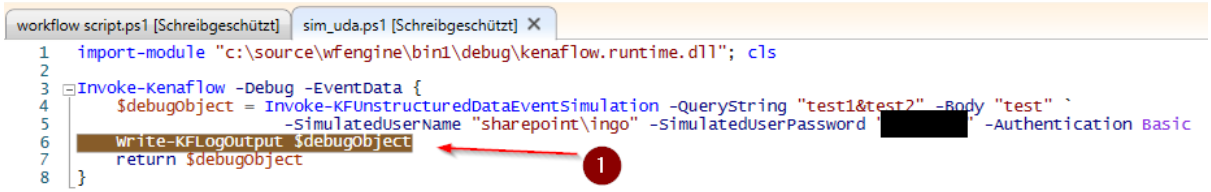
In a normal (time based) workflow run the workflow script does nothing because `$eventData` is empty. But when you debug the test script `debug-unstructured-data.ps1` you have a valid `$eventData` object and get all the request data in the output window.

You are able to debug into the test script `debug-unstructured-data.ps1` and also to debug into the script block that contains cmdlet `Invoke-KFUnstructuredDataEventSimulation` in the above example.

You can have a look at the object, that `Invoke-KFUnstructuredDataEventSimulation` creates. Lets have a look at this screenshot:



Not the test script is currently open but the workflow script. See the line marked with (1) ? The debugger has stopped there. The block marked with (2) shows the object that was created by `Invoke-KFUnstructuredDataEventSimulation`. The corresponding test script is this:



There we catch the object and write it line marked with (1) to the log using *`Write-KFLogOutput`*.

Here is an example test script for a SharePoint Remote Event:

```
    Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

    Invoke-Kenaflow -EventData {
            $debugObject =
                    Invoke-KFSharePointEventSimulation `
                            -ItemId 1 `
                            -EventType ItemUpdated `
                            -SimulatedUserName "sharepoint\ingo" `
                            -Data @{Title = "kenaflow is cool!"}

            Write-KFLogOutput $debugObject

            return $debugObject
    }
```

The log message in the output window is:

```
    VERBOSE: kf[normal  ]: {"EventSource":1,"EventData":"\r\n<s:Envelope xmlns:s=\
↪"http://schemas.xmlsoap.org/soap/en
    velope/\">\r\n\t<s:Body>\r\n\t\t<ProcessOneWayEvent xmlns=\"http://schemas.
↪microsoft.com/sharepoint/remoteapp/\">\r\n
    \t\t\t<properties xmlns:i=\"http://www.w3.org/2001/XMLSchema-instance\">\r\n\t\
↪t\t\t<AppEventProperties i:nil=\"true\
    "/>\r\n\t\t\t\t<ContextToken/>\r\n\t\t\t\t<CorrelationId>00000000-0000-0000-
↪0000-000000000000</CorrelationId>\r\n\t\t
    \t\t<CultureLCID>1031</CultureLCID>\r\n\t\t\t\t<EntityInstanceEventProperties␣
↪i:nil=\"true\"/>\r\n\t\t\t\t<ErrorCode/
    >\r\n\t\t\t\t<ErrorMessage/>\r\n\t\t\t\t<EventType>ItemUpdated</EventType>\r\n\
↪t\t\t\t<ItemEventProperties>\r\n\t\t\t
    \t\t<AfterProperties xmlns:a=\"http://schemas.microsoft.com/2003/10/
↪Serialization/Arrays\">\r\n\t\t\t\t\t\t<a:KeyValu
    eOfstringanyType>\r\n\t\t\t\t\t\t<a:Key>Title</a:Key>\r\n\t\t\t\t\t\t
↪<a:Value i:type=\"b:string\" xmlns:b=\"http:
    //www.w3.org/2001/XMLSchema\">kenaflow is cool!</a:Value>\r\n\t\t\t\t\t</
↪a:KeyValueOfstringanyType>\r\n\t\t\t\t</
    AfterProperties>\r\n\t\t\t\t<AfterUrl i:nil=\"true\"/>\r\n\t\t\t\t
↪<BeforeProperties xmlns:a=\"http://schemas.micr
    osoft.com/2003/10/Serialization/Arrays\"/>\r\n\t\t\t\t<BeforeUrl/>\r\n\t\t\t\
↪t\t<CurrentUserId>6</CurrentUserId>\r\
    n\t\t\t\t<ExternalNotificationMessage i:nil=\"true\"/>\r\n\t\t\t\t
↪<IsBackgroundSave>false</IsBackgroundSave>\r\n\
    t\t\t\t<ListId>2ad9bcc4-9f8b-474c-b485-56944d68baca</ListId>\r\n\t\t\t\t
↪<ListItemId>1</ListItemId>\r\n\t\t\t\t<
    ListTitle>Set-KF-Perm-Test</ListTitle>\r\n\t\t\t\t<UserDisplayName>Ingo</
↪UserDisplayName>\r\n\t\t\t<UserLoginNa
    me>i:0#.w|sharepoint\\ingo</UserLoginName>\r\n\t\t\t<Versionless>false</
↪Versionless>\r\n\t\t\t<WebUrl>https:/
    /intranet13.sharepoint.farm</WebUrl>\r\n\t\t\t</ItemEventProperties>\r\n\t\t\
↪t\t<ListEventProperties i:nil=\"true\"
    />\r\n\t\t\t<SecurityEventProperties i:nil=\"true\"/>\r\n\t\t\t<
↪<UICultureLCID>1031</UICultureLCID>\r\n\t\t\t<We
    bEventProperties i:nil=\"true\"/>\r\n\t\t\t</properties>\r\n\t\t</
↪ProcessOneWayEvent>\r\n\t</s:Body>\r\n</s:Envelope>
    \r\n","OutputFormat":0}
    VERBOSE: kf[normal  ]: Excluding users: |app@sharepoint; sharepoint\kenaflow
    VERBOSE: kf[normal  ]: connecting to SharePoint...
    VERBOSE: kf[normal  ]: current user: i:0#.w|sharepoint\kenaflow (Microsoft.
↪SharePoint.Client.UserIdInfo)
    VERBOSE: kf[normal  ]: connected
    VERBOSE: kf[normal  ]: Last action timestamp not found
    VERBOSE: kf[normal  ]: Set default config variables
    VERBOSE: kf[normal  ]: Preparing query...
```

(continues on next page)

(continued from previous page)

```
    VERBOSE: kf[normal  ]: current item: #1 // 56b2704e-78a9-4bb3-94e6-8eb6914ea3d9
```

You can see a "native" looking SharePoint remote event XML in the log.

The *kenaflow* loads the workflow script an initializes it for the item specified in the remote event.

11.3.1.4.1 Debugging "Unstructured Data" Remove Event Receivers with Output Data

Please read the corresponding section in article *Remote Event* first!

Workflow with an remote event of type "unstructured data" can return output data. In normal execution this is send back in the http connection as response.

In the simulation - for debugging - you receive this data as result of `Invoke-Kenaflow`.

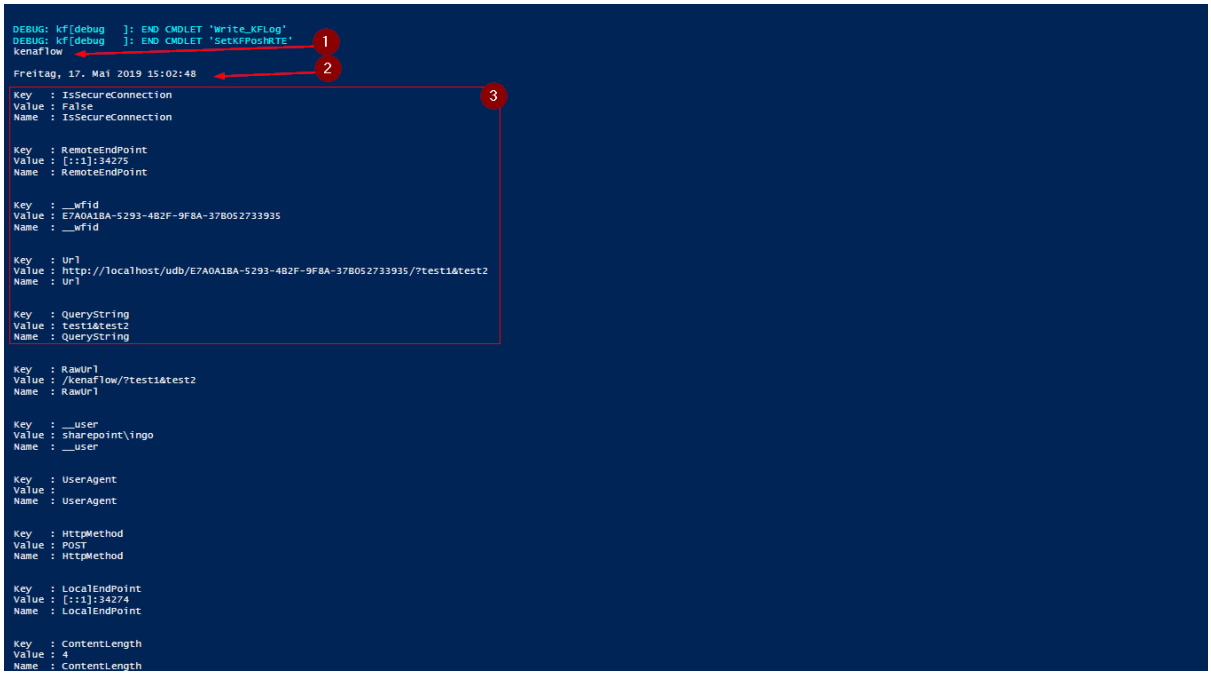
Example:

```
    import-module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

    $output = Invoke-Kenaflow -Debug -EventData {
        Invoke-KFUnstructuredDataEventSimulation -QueryString "test1&test2" -Body
↪"test" `
                    -SimulatedUserName "sharepoint\ingo" -SimulatedUserPassword
↪"kr@ftwerk" -Authentication Basic `
                    -OutputFormat Unconverted
    }

    $output.Test
    $output.currentDate
    $output.Evt
```

You get as result:



(1) is the output of `$output.Test`(2) is the output of `$output.currentDate`(3) is (part of) the output of `$output.Evt`

Or you request a JSON string with `-OutputFormat JsonString` as you would receive it through a real web request:

```
    [{"CurrentDate":"2019-05-16T17:02:52.2971184+02:00","Evt":{"IsSecureConnection
↪":false,"Referrer":null,"__wfid":"E7A0A1BA-5293-4B2F-9F8A-37B052733935","Url":
↪"http://localhost/udb/E7A0A1BA-5293-4B2F-9F8A-37B052733935/?test1&test2",
↪"QueryString":"test1&test2","__user":"sharepoint\\ingo","UserAgent":null,
↪"HttpMethod":"POST","LocalEndPoint":"[::1]:56018","ContentLength":4,
↪"IsAuthenticated":false,"UserHostAddress":"[::1]:56018","RemoteEndPointIP":"::1",
↪"AcceptTypes":null,"RemoteEndPointPort":"56019","RemoteEndPoint":"[::1]:56019",
↪"IsLocal":true,"ContentType":null,"KeepAlive":true,"__userAuthType":"Basic","__
↪userIsAuthenticated":true,"ProtocolVersion":"1.1","UserLanguages":null,"RawUrl":
↪"/kenaflow/?test1&test2","UserHostName":"localhost:56018","__body":"test",
↪"Headers":{"Connection":"Keep-Alive","Host":"localhost:56018","Expect":"100-
↪continue","Content-Length":"4"},"ContentEncoding":"System.Text.
↪SBCSCodePageEncoding"},"Test":"kenaflow"}]
```

Or you request a PowerShell CliXML string with `-OutputFormat ClixmlString` as you would receive it through a real web request:

```
    <Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04
↪">
      <Obj RefId="0">
        <TN RefId="0">
          <T>System.Collections.Generic.List`1[[System.Management.Automation.
↪PSObject, System.Management.Automation, Version=3.0.0.0, Culture=neutral,␣
↪PublicKeyToke
    n=31bf3856ad364e35]]</T>
          <T>System.Object</T>
        </TN>
        <LST>
          <Obj RefId="1">
            <TN RefId="1">
              <T>System.Collections.Hashtable</T>
              <T>System.Object</T>
            </TN>
            <DCT>
              <En>
                <S N="Key">CurrentDate</S>
                <DT N="Value">2019-05-16T17:04:19.0350126+02:00</DT>
              </En>
              <En>
                <S N="Key">Evt</S>
                <Obj N="Value" RefId="2">
                  <TNRef RefId="1" />
                  <DCT>
                    <En>
                      <S N="Key">IsSecureConnection</S>
                      <B N="Value">false</B>
                    </En>
                    <En>
                      <S N="Key">Referrer</S>
                      <Nil N="Value" />
                    </En>
                    <En>
                      <S N="Key">__wfid</S>
                      <S N="Value">E7A0A1BA-5293-4B2F-9F8A-37B052733935</S>
                    </En>
                    <En>
                      <S N="Key">Url</S>
                      <S N="Value">http://localhost/udb/E7A0A1BA-5293-4B2F-9F8A-
↪37B052733935/?test1&amp;test2</S>
                    </En>
                    <En>
                      <S N="Key">QueryString</S>
```

(continues on next page)

(continued from previous page)

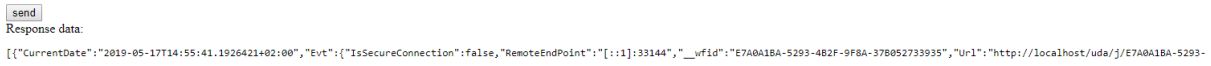
```
                    <S N="Value">test1&amp;test2</S>
                  </En>
```

(This is only a part of the result CliXML string!)

If you want to test such an request from JavaScript you could create a simple HTML page with an jQuery based AJAX request:

```
    <html>
    <script src="jquery.js"></script>
    <script type="text/javascript">
    function send(){
            $.post('http://localhost:80/uda/j/E7A0A1BA-5293-4B2F-9F8A-37B052733935/
↪',JSON.stringify({
                                        "test":"kenaflow is cool!"})).done(
                                  function(data, State) {
                                        $('#response').html('Response data: ');
                                        $('#data').html(data);
                                  });
    }
    </script>
    <body>
    <input type="button" onclick="javascript:send();return false" value="send"/>
    <div id="response"></div>
    <pre id="data"></pre>
    </body>
    </html>
```

This is the result in the browser:



11.4 Libraries

You can specifiy PowerShell files as libraries at the global configuration level and in each workflow configuration file.

These libraries are executed immediately BEFORE each workflow script or pre-query script.

If specified first the global library is executed and than second the local library.

Global library configuration (kenaflow.conf):

```
20      <entry key="MaxExecutionTime" type="System.Int32" default="600"></entry>
21      <entry key="CustomLibrary" type="System.String" default="" comment="relative to program folder"></entry>
22    </section>
```

The path is relative to the program directory (e.g. `C:\Program Files\kenaflow`).

Local library in workflow configuration file `_wfconfig.ps1`:

```
1   @{
2       Version = "0.3";
3       Ignore = $false;
4       Name = "wf4b";
5       Platform = "spo";
6       Id = "                                    ";
7       CustomLibrary="test.psl";
```

The path is relative to the workflow folder.

The use of global variables is possible:

```
$global:value = 1
function HelloKenaflow() {
        Write-KFLog "Hello kenaflow: $global:value"
        $global:value++
}
```

For each list element that *in the context of a workflow execution* of is processed in a workflow script, you can use this function `Hellokenaflow`, whereby the contents of the global variable $`$global:value` is increased by 1.

When processing of this workflow is complete, the PowerShell session and the global variables are discarded.

!! Both library files do not operate within an SharePoint or email context. Parameters are not provided in this script.

11.5 Workflow Lists

For each workflow, several lists located in Sharepoint can be specified via the workflow configuration and used for workflow execution.

All of the lists below except the "Status List" are optional and do not need to be used.

- *Config List*
- *Data List*
- *History List*
- *State List*
- *Error List*

11.5.1 Config List

The "Workflow Config" list is a SharePoint list in which configuration values for a workflow can be stored.

The list consists of simple key-value pairs.

Workflows can share a "Workflow Config" list.

There are command line commands for kenaflow.exe to create, hide and display the "Workflow Config"-list.

See also the program switches

- *`--createconfiglist`*
- *`--showconfiglist`*
- *`--hideconfiglist`*

The config values from the workflow configuration file `_wfconfig.ps1` are written to the SharePoint list when using `--createconfiglist`. You can *add* more values to `_wfconfig.ps1` and write them to SharePoint using `--createconfiglist` again. **But** already existing values in SharePoint are not overwritten! You could remove them from SharePoint to allow `--createconfiglist` to write the updated values to the SharePoint list.

You can use *`--backupconfiglist`* to export the current config values from the SharePoint list.

11.5.2 Data List

The "Workflow Data" list is a SharePoint list where workflow scripts can persist data for later use.

For example, the `Set-KFData` and `Get-KFData` cmdlets can be used to read and write data relative to the current list item. (Applies only to List Workflows and State Machine Workflows).

There are further cmdlets: `Set-KFItemData` and `Get-KFItemData`.

This data is stored in SharePoint in serialized form.

It will be serialized in two formats:

1. PowerShell-Serializing (XML)
2. JSON-Serializing

When reading the data with `Get-KFData`, only the PowerShell serialization is read.

JSON serialization may be used to use the workflow data in Web applications, where the Workflow Data List is queried using the SharePoint REST API.

Several Workflows can share a workflow data list.

There are commands for kenaflow.exe, to create, hide and display the "Workflow Data" list.

Therefore look up the program switches

- *--createdatalist*
- *--hidedatalist*
- *--showdatalist*

11.5.3 History List

The "history" list can be used to have logging information written out of a workflow script in a dedicated SharePoint list.

This is done using the `Write-KFHistory` cmdlet.

Several workflows can share a history list

There are command line commands for kenaflow.exe to create, hide and display the "history"-list.

See program switch

- *--createhistorylist*
- *--hidehistorylist*
- *--showhistorylist*

11.5.4 State List

The status list is only relevant for "State Machine" workflows.

The status list is a SharePoint list that contains a record for each workflow step.

The status code is central. It is also used in the workflow configuration and it uniquely identifies a status.

In addition, a description of the workflow step can be provided.

The status list is linked to the SharePoint list on which the workflow operates.

The SharePoint list on which the workflow operates has a lookup column that points to the status list.

Multiple workflows can share a status list.

There are command line commands for kenaflow.exe to create, hide and display the status list and to add it to the SharePoint list where the workflow is working.

See program switches:

- `--createstatelist`
- `--hidestatelist`
- `--showstatelist`
- `--addstatefield`

11.5.5 Error List

If configured, the workflow saves errors and warnings in the "Error List".

- `--createerrorlist`
- `--showerrorlist`
- `--hideerrorlist`

11.6 Locking

kenaflow was designed to execute a huge number of workflows. Therefore it is possible to scale it over several servers.

For that reason it is necessary to prevent one workflow to run at the same time on multiple *kenaflow* servers.

This requires locking.

During execution workflows are locked by creating a 0-byte lock file in a defined folder or network share. The lock file is removed after the processing of the workflow is finished.

kenaflow for SharePoint is able to run multiple workflows on the same SharePoint list. (This is only related to *list workflows* and *state machine workflows*!)

For that reason it is necessary to prevent multiple workflows - that could run in parallel - to process the same SharePoint list item concurrently.

This would lead - at least - in "version conflicts".

Therefore *kenaflow* locks each currently processed SharePoint list item using a 0-byte lock file in a defined folder or network share. This lock file is removed after the processing of the item is finished.

It could be necessary to access other lists in workflows than the workflow list. Or access a list in a SharePoint site workflow. Then you have lock SharePoint items manually to prevent parallel write operations.

Therefore you can use these cmdlets:

- `Lock-KFItem`
- `Test-KFItemLock`
- `Unlock-KFItem`

With them you are able to lock SharePoint list items.

!! The lock does not affect other systems than *kenaflow*! It is not visible to SharePoint / users!

To orchestrate workflows it is sometimes not enought to leverage SharePoint list item locks. But the developer of a workflow maybe need to prevent processing of one workflow when another runs.

Therefore *kenaflow* offers a global locking mechanism by using custom keys for creating, releasing and testing locks.

Therefore we create there cmdlets:

- *Lock-KFProcess*
- *Test-KFProcessLock*
- *Unlock-KFProcess*

When using this custom "process" locks *kenaflow* creates 0-byte lock files in a defined folder or network share. The lock file is removed after the processing of the workflow is finished. When other workflows try to create a lock with an already used key they get an error and have to wait or postpone.

The lock file location is configured in the setting "lockDir" in the global configuration section *"Locking"*.

With this setting multiple *kenaflow* instances or servers can be configured to use the same place for lock files.

11.7 Links

Sometimes it is required to simplify workflows as much as possible. E.g. to reduce the amount of time that users have to invest in business processes. Some of these processes do not require complex forms to be filled out. Sometimes it's enough to give a simple answer to a simple question. Yes or no.

Therefore we created a very useful mechanism for *kenaflow*: "rich links" we call it.

They are generated with the cmdlet *Get-KFLink*.

With it you can generate a complex ("rich") link with **encrypted** data in the query string protion of the URL.

The links contains:

- an optional validity timestamp and
- an optional metadata object^^.

The links can work asynchronous or synchronous.

"Asynchronous" means, that the user gets an simple answer in the browser that the request was received. The user does not get a result of the action itself.

"Synchronous" means, that the user gets a result from the workflow - as long as the workflow was fast enough to answer. This is because the workflow execution itself is always "asychronous" because of the architecture of *kenaflow*. It cannot be guaranteed to process a remote event in a very short period of time. Because remote events are queued and if the queue is long it may take a while until a remote event is processed.

The generated link can be send to a user e.g. in an email using out template engine (*Templates*) with *Send-KFMail*.

When the user clicks the link a browser opens.

In case of both "asynchronous" link remote events the user does only receive a generic answer in the browser that tells that the link event was received by *kenaflow*. (If the link is expired the user gets an information about the expiration.)

If the link is expired - if `-ValidUntil` is set - the user get an immediate expiration answer. The HTML for this answer can be configured in the global configuration of *kenaflow*. It's currently not possible to configure this answer specific to the workflow because the workflow configuration `_wfconfig.ps1` is not loaded at the point in time where the link remote event is received. If you want to implement you own expiration notice you can do that by your own. E.g. you use `-Metadata` or `Get-KFLink` and send the validity timestamp this way to the workflow.

In case of both "asynchronous" and "synchronous" link remote events the user does not get informed that the link did not cause any action (because of invalidity).

As already mentioned it is possible to send custom data to the workflow in a link remote event using parameter `-Metadata`. It accepts a PowerShell Hashtable object that is serialized and encrypted in the link and is later available in the workflow as part of the `$eventData` object in property `__metadata`.

In the workflow the link remote events object in `$eventData` has the following properties:

- `__wfid` : The workflow id.

- `__itemid` : The item id the link remote event belongs to.
- `__valid` : The "valid until" timestamp.
- `__withAnswer` : `true` if its a link remote event with required answer.
- `__metadata` : The data ob the Hashtable specified in parameter `-MetaData` of `Get-KFLink`.

Using `Get-KFLink` it is possible to generate links for different workflows. Therefore you specify the workflow id with `-WorkflowId`. BUT BE CAREFUL because the other workflow gets the ID of the currently processed item and will use this ID to process the item. If for example the other workflow is connected to another list the ID may point to the wrong item!

You can also specify a different list item than the currently processed one using `-Item`.

You can simulate a link remote event for debugging using a script like this:

```
    Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

    Invoke-Kenaflow -EventData {
        Invoke-KFLinkEventSimulation -NTLM -SimulatedUserName "sharepoint\ingo" \`
          -SimulatedUserPassword "kr@ftwerk" -ItemId 1 -WithAnswer -Metadata @{
↪"kenaflow"="cool"}
    }
```

11.8 Workflow Safety

We have installed several devices to ensure the workflow execution.

11.8.1 Maximum Execution Time

(to be written)

11.8.2 Script Run Timeout

- *Set-KFScriptRunTimeout*
- *Get-KFScriptRunTimeout*

11.9 Version Upgrade

kenaflow is constantly being further developed. We have 1000 ideas that we want to implement. From time to time we will publish new major versions, which represent a milestone in the further development.

But there will be not only this kind of further development. Of course, we are constantly working on improving the quality of the software. This includes bug fixes and new feature details.

And something else... *kenaflow* uses a number of third-party libraries. Perhaps the most important is the "SharePoint PnP PowerShell" module. The PowerShell library developed by the Community & Microsoft is integrated into *kenaflow*. These and other libraries are also adapted and changed over time. With new versions of *kenaflow* we integrate new versions of these libraries.

An overview of the third-party libraries used by *kenaflow* can be found in the section *Open Source Licenses*.

All these points give reason for changes. We try to keep these changes transparent for our users, but we don't always succeed.

For example, the SharePoint PnP PowerShell development team replaced the `Execute-PnpQuery` cmdlet with `Invoke-PnpQuery` in version 2.24.1803. As a result, all scripts are no longer compatible with the new version.

(Although there is an "alias" here to keep `Execute-PnPQuery` working.) If the updated library is transferred to *kenaflow*, the workflows scripts will also need to be modified, e.g. to replace discontinued cmdlets.

In this section, we're gathering information to simplify the upgrade.

! Please monitor the change log of "SharePoint PnP PowerShell" at ! https://github.com/SharePoint/PnP-PowerShell/blob/master/CHANGELOG.md

11.9.1 Upgrade Paths

- *1.0 => 2.0*

11.9.2 1.0 => 2.0

- Make sure you use `Invoke-PnPQuery` instead of `Execute-PnPQuery`!

11.10 Permissions

11.10.1 SharePoint Permission Sets

In the workflow configuration file, you can specify four permission sets that can be quickly and easily assigned to a user using the `Add-KFPermission`, `Remove-KFPermission`, and `Update-KFPermission` cmdlets.

We have compiled 6 meaningful SharePoint permission set that are often used. These follow the well-known CRUD model (Create, Read, Update, Delete). We have added "M" for "Manage" and "A" for "Approve".

Users or groups that have the corresponding permission set can perform very special tasks.

For example, anyone assigned "Create" may create elements, but not automatically read them. Accordingly, a user with "Read" permission may not automatically create elements.

You can specify names for the 6 permission sets in the workflow configuration file.

With the program command `--createpermissionsets` these permission sets are created in SharePoint and equipped with rights ("Base Permissions") according to our specifications.

Here is an overview of how the permission sets are defined:

Create:

- Add list Items (AddListItems)
- Show List Items (ViewListItems)
- Create Alerts (CreateAlerts)
- View Pages (ViewPages)
- Open (Open)

Read:

- View list Items (ViewListItems)
- Open List Items (OpenItems)
- Show Version (ViewVersions)
- Show Pages (ViewPages)
- Open (Open)

Update:

- Edit List Items (EditListItems)

- Show List Items (ViewItems)
- View Pages (ViewPages)
- Open (Open)

Delete:

- Delete List Items (DeketeListItems)
- View List Items (ViewItems)
- Show Versions (ViewVersions)
- Delete Versions (DeleteVersions)
- Use APIs (UseRemoteAPIs)
- View Pages (ViewPages)
- Open (Open)

Manage:

- Edit list items (EditListItems)
- View list items (ViewItems)
- Approve list items (ApproveItems)
- Open list Items (OpenItems)
- Browse directories (BrowseDirectories)
- Show Versions (ViewVersions)
- Enumerate Permissions (EnumeratePermissions)
- Manage Permissions (ManagePermissions)
- Browse User Infos (BrowseUserInfo)
- Open (Open)

Approve:

- View list items (ViewItems)
- Approve list items (ApproveItems)
- Open list Items (OpenItems)
- Show Versions (ViewVersions)
- Browse directories (BrowseDirectories)
- Enumerate Permissions (EnumeratePermissions)
- Browse User Infos (BrowseUserInfo)
- Open (Open)

11.11 Remote Event

11.11.1 Introduction

“Remove Events are a setup of SharePoint. We use this name for the general ability of *kenaflow* to receive data from other systems and then execute workflows. - Workflows always consist of PowerShell scripts. This means that the data from other systems - the “remote events” - will be passed on to the scripts. Based on this data, actions can then be performed by the scripts. The workflow developer is free to use the data.

There are the following variants of “Remove Events”:

- SharePoint Remove Events: when **adding** or **updating** an SharePoint list item
- custom data
- unstructured data

Here is a table which *Workflow Types* can receive which types of remote events.

| | SharePoint Remote Event | custom data | unstructured data | | - |-|-|-| | List Workflow | yes | yes | - | | State Machine Workflow | yes | yes | - | | Site Workflow | - | - | yes | | PowerShell Workflow | - | - | yes | | Email Workflow | - | - | - |

11.11.2 The Execution of Remote Events

Remote Events most likely need to be executed really soon after they are raised.

Besides the normal *time/queue based* workflow execution *kenaflow* has a separate event exection machanism that we call “Remote Event Runner”. (Our naming “RER” means both “Remote Event Receiver” and “Remote Event Runner”!)

Remove Events always come by “http(s)”. From SharePoint of from third party systems.

There is one *kenaflow* process that hosts *all* http(s) endpoints of the *kenaflow* instance. It is started by the “main mode” (`kenaflow.exe --run`). It’s mode parameter is `--rer` (“Remote Event Receiver”). It receives all events and stores them in small files in the “queue folder”.

These files are encrypted. But you can disable encryption in the global configuration, *Section “RER”*.

By default there is at least one process per SharePoint version (2013, 2016, 2019, Online) and one for pure PowerShell workflows. This processes monitor the queue directory. They are started by the “main mode” process of *kenaflow* with the mode parameters:

- *--rerrunnersp2013*
- *--rerrunnersp2016*
- *--rerrunnersp2019*
- *--rerrunnerspo*
- *--rerrunnerposh*

Each one of them monitors the “queue folder” and executes the events for all workflows that belong to the runner’s SharePoint version or “platform”.

The runner process decrypts the events and executes it.

It knows where to find the SharePoint web and list (in case of an event coming *for* a SharePoint workflow). It creates a PowerShell runtime universe in a nutshell and executes the workflow script in it. Than the PowerShell universe ist destroyed before the next event is processed.

You can have multiple runners per platform - or none. You can configure it in the global configuration.

E.g. if you have an on-premises SharePoint 2016 and no other SharePoint environment than you can disable all runners except the SharePoint 2016 runner. But for SharEPoint 2016 you can specify 5 runners to process events faster.

```
        <entry key="rerRunnerSP2013" type="System.Int32" default="1">0</entry>
        <entry key="rerRunnerSP2016" type="System.Int32" default="1">5</entry>
        <entry key="rerRunnerSP2019" type="System.Int32" default="1">0</entry>
        <entry key="rerRunnerSPO" type="System.Int32" default="1">0</entry>
        <entry key="rerRunnerPOSH" type="System.Int32" default="1">0</entry>
```

11.11.3 Debugging of Remote Events

Debugging of remote is easy. Please read the article *Debugging*.

11.11.4 SharePoint Remove Events

External systems such as *kenaflow* can subscribe event notifications from SharePoint.

kenaflow registers - if configurated - subscripts for “Item Added” and “Item Updated” events. This is only possible in context of list items.

SharePoint - 2013, 2016, 2019 and *ONLINE* - sends http(s) requests to the subscriber. This contains XML.

!! You can only use http and port 80 or https and port 443. This is a limitation of SharePoint.

SharePoint remote events are send in SOAP envelops. Here is a sample:

```
    <s:Envelope xmlns:s=""http://schemas.xmlsoap.org/soap/envelope/"">
        <s:Body>
            <ProcessOneWayEvent xmlns=""http://schemas.microsoft.com/sharepoint/
↪remoteapp/"">
                <properties xmlns:i=""http://www.w3.org/2001/XMLSchema-instance"">
                    <AppEventProperties i:nil=""true""/>
                    <ContextToken/>
                    <CorrelationId>00000000-0000-0000-0000-000000000000</
↪CorrelationId>
                    <CultureLCID>1031</CultureLCID>
                    <EntityInstanceEventProperties i:nil=""true""/>
                    <ErrorCode/>
                    <ErrorMessage/>
                    <EventType>ItemAdded</EventType>
                    <ItemEventProperties>
                        <AfterProperties xmlns:a=""http://schemas.microsoft.com/
↪2003/10/Serialization/Arrays"">";
                            <a:KeyValueOfstringanyType>
                                <a:Key>Title</a:Key>
                                <a:Value i:type=""b:string"" xmlns:b=""http://www.
↪w3.org/2001/XMLSchema"">Hello kenaflow!</a:Value>
                            </a:KeyValueOfstringanyType>";
                        </AfterProperties>
                        <AfterUrl i:nil=""true""/>
                        <BeforeProperties xmlns:a=""http://schemas.microsoft.com/
↪2003/10/Serialization/Arrays""/>
                        <BeforeUrl/>
                        <CurrentUserId>1</CurrentUserId>
                        <ExternalNotificationMessage i:nil=""true""/>
                        <IsBackgroundSave>false</IsBackgroundSave>
                        <ListId>1</ListId>
                        <ListItemId>77c52d58-bbdb-41ae-975f-eef13fd42a21</
↪ListItemId>
                        <ListTitle>Custom List</ListTitle>
                        <UserDisplayName>Ingo</UserDisplayName>
                        <UserLoginName>sharepoint\ingo</UserLoginName>
                        <Versionless>false</Versionless>
```

(continues on next page)

(continued from previous page)

```
                    <WebUrl>https://intranet19.sharepoint.farm</WebUrl>
                </ItemEventProperties>
                <ListEventProperties i:nil=""true""/>
                <SecurityEventProperties i:nil=""true""/>
                <UICultureLCID>1031</UICultureLCID>
                <WebEventProperties i:nil=""true""/>
            </properties>
        </ProcessOneWayEvent>
    </s:Body>
</s:Envelope>
```

11.11.5 Custom Data

This remote event type is also related to sharepoint list item.

It is used to trigger the workflow to execute a script on the given item.

The incoming data can be send with http method "GET" by using the query string URL portion for the data. Or method "POST" where the data is in the http request stream. If a query string portion is specified this will be uses instead of the request stream!!!

The endpoint URL looks like this:

`https://incoming.kenaflow.com/<instance_guid>/cd<X>/<workflow_guid>/`

The `<instance_guid>` is the guid of the *kenflow* instance. This URL part can be disabled in the global configuration.

The `<X>` stands for one of these alternatives:

- `A` for "anonymous"
- `N` from NTLM authentication
- `B` for basic authentication

In case of the autentication alternatives the sender of data need to authenticate itself. *kenaflow* does verify the identity against Active Directory, but it does not *authorize* a request. It just sends the information about the user to the workflow script. The script need to decide wether to process the "remove event" or not.

The workflow script has a global parameter `$eventData` that contains the remove event object in case of workflow execution in context of an received remote event. Otherwise this parameter is `$null`

This is an object. In case of "custom data" it has the following properties:

- `__user` : Login name of the user or `null`
- `__userIsAuthenticated` : bool that indicates the authentication state. This property does not exist if `__user` is `null`.
- `__userAuthType` : the used authentication type. This property does not exist if `__user` is `null`.

The data send to the workflow need to be in JSON format:

```
{ "__wfid" : "<workflow_guid>", "__itemid" : <item_id>, <custom_data> }
```

As you can see here only to parameters are required: `__wfid` and `__itemid`. They need to be specified.

When sending data to a SharePoint *list* or *state machine* workflow you need to send the ID of the related SharePoint list item to the workflow using **querystring** parameter `__itemid`.

When sending data to a SharePoint *list* or *state machine* workflow that has more than 1 connected list you need to specify the corresponding list using **querystring** parameter `__list`. It names the list by its *title*.

When sending data to a SharePoint *list* or *state machine* workflow that has more than **1 connected web** you need to specify the corresponding web using **querystring** parameter `__web`. It names the web by its *URL*.

```
    { "__wfid" : "<workflow_guid>", "__itemid" : <item_id>,
      "__list": "Custom List", "__web": "https://intranet.sharepoint.farm/sites/
↪kenaflow",
      <custom_data> }
```

11.11.6 Unstructured Data

This belongs to all types of *SharePoint* workflows and to *PowerShell* workflows.

Unlike "Custom Data" remote events, "Unstructured Data" remote events allow you to send any data to the workflow. They don't have to be sent in JSON format and don't have to contain item IDs or similar.

The entire http request is given to the workflow in the form of an object that then contains, for example, the property "__body" in which the entire request stream is stored as string. All other request data that is accessible for *kenaflow* is passed on to the workflow.

The endpoint URL looks like this:

`https://incoming.kenaflow.com/<instance_guid>/ud<X>/<workflow_guid>/`

Data have to be sent with http method "POST"! The query string is ignored but passed to the workflow script.

The `<instance_guid>` is the guid of the *kenflow* instance. This URL part can be disabled in the global configuration.

The `<X>` stands for one of these alternatives:

- `A` for "anonymous"
- `N` from NTLM authentication
- `B` for basic authentication

In case of the autentication alternatives the sender of data need to authenticate itself. *kenaflow* does verify the identity against Active Directory, but it does not *authorize* a request. It just sends the information about the user to the workflow script. The script need to decide wether to process the "remove event" or not.

The workflow script has a global parameter `$eventData` that contains the remove event object in case of workflow execution in context of an received remote event. Otherwise this parameter is `$null`

This is an object. In case of "custom data" it has the following properties:

- `__user` : Login name of the user or `null`
- `__userIsAuthenticated` : bool that indicates the authentication state. This property does not exist if `__user` is `null`.
- `__userAuthType` : the used authentication type. This property does not exist if `__user` is `null`.

You get an PowerShell Hashtable object of the following structure:

```
Name                           Value
----                           -----
__body                         test
__user                         sharepoint\ingo
__userAuthType                 Basic
__userIsAuthenticated          True
__wfid                         E7A0A1BA-5293-4B2F-9F8A-37B052733935
AcceptTypes
ContentEncoding                System.Text.SBCSCodePageEncoding
ContentLength                  4
ContentType
Headers                        {Connection, Host, Expect, Content-Length}
HttpMethod                     POST
IsAuthenticated                False
IsLocal                        True
```

(continues on next page)

(continued from previous page)

```
    IsSecureConnection              False
    KeepAlive                       True
    LocalEndPoint                   [::1]:40969
    ProtocolVersion                 1.1
    QueryString                     {qs}
    RawUrl                          /kenaflow/?qs=test1
    Referrer
    RemoteEndPoint                  [::1]:40970
    RemoteEndPointIP                ::1
    RemoteEndPointPort              40970
    Url                             http://localhost/udb/E7A0A1BA-5293-4B2F-9F8A-..
↪.
    UserAgent
    UserHostAddress                 [::1]:40969
    UserHostName                    localhost:40969
    UserLanguages
```

When sending data to a SharePoint *list* or *state machine* workflow you need to send the ID of the related SharePoint list item to the workflow using **querystring** parameter `__itemid`.

When sending data to a SharePoint *list* or *state machine* workflow that has more than 1 connected list you need to specify the corresponding list using **querystring** parameter `__list`. It names the list by its *title*.

When sending data to a SharePoint *list* or *state machine* workflow that has more than **1 connected web** you need to specify the corresponding web using **querystring** parameter `__web`. It names the web by its *URL*.

11.11.7 Receiving Data from a Workflow

You can receive data from a workflow that you call by using a web request.

This is currently only valid for "Unstructured Data" remove events, send to SharePoint *site* workflows or PowerShell workflows.

To receive data from a workflow you use `/w` (for PowerShell CliXML serialized return data) or `/j` for JSON serialized return data or `/b` for binary return data in the URL between `/uda` and the workflow id.

Example: *kenaflow* is running locally. There is a workflow with this config (`_wfconfig.ps1`):

```
    @{
        Version = "4.0";
        Enabled = $true;
        Type = "SHAREPOINT";
        SubType = "SITE";
        Name = "unstructured-rer-test-with-wait";
        Platform = "sp2013";
        Id = "E7A0A1BA-5293-4B2F-9F8A-37B052733935";
        TBE = -1;
        ErrorHandlingScript = "errorHandling.ps1";
        spUser = "env\kenaflow";
        spPwd = "kr@ftwerk";
        Web = "https://intranet13.sharepoint.farm";
        RER = $true;
        Script = "script.ps1";
    }

    if(!$kenaflow){import-module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Test-KFConfig;exit}
```

With this workflow script:

```
    param($wf, $web, $config, $eventData)

    if($wf-eq$null){import-module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Invoke-Kenaflow;exit}

    return @{Test="kenaflow"; CurrentDate = [DateTime]::Now; Evt = $eventData}
```

From a fresh powershell console I execute the following web request:

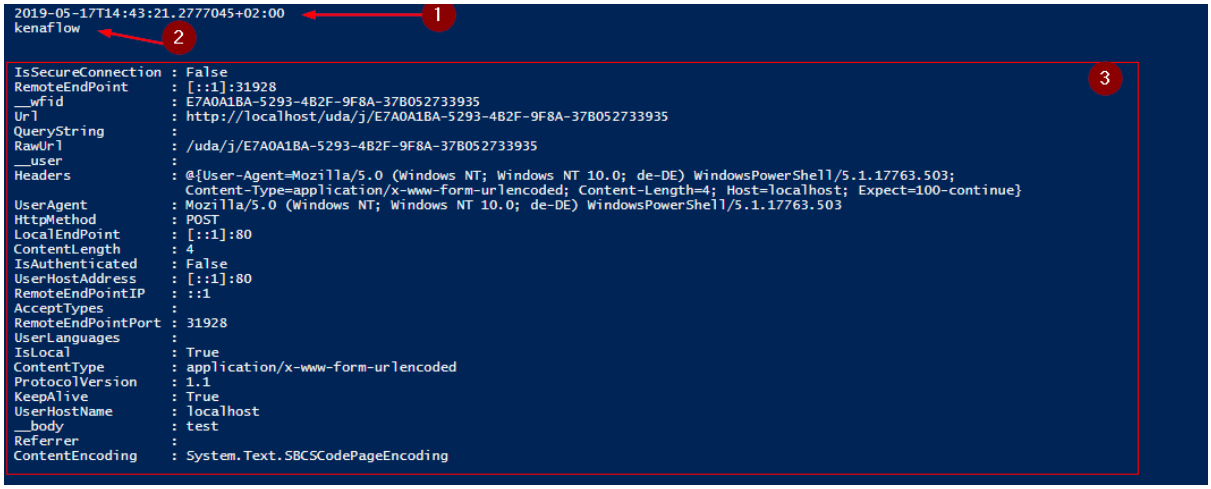
```
    $result = (Invoke-WebRequest "http://localhost:80/uda/j/E7A0A1BA-5293-4B2F-
↪9F8A-37B052733935" -Body "test" -Method "POST")
```

See the "j/" in it?

I deserialize the response body with this and look at the two result objects

```
    $jsonResult = ConvertFrom-Json ([System.text.encoding]::utf8.getstring($result.
↪Content))
    $jsonResult.CurrentDate
    $jsonResult.Test
    $jsonResult.Evt
```

Thats the result:



(1) is `$jsonResult.CurrentDate`(2) is `$jsonResult.Test`(3) is `$jsonResult.Evt` - the actual event object that is passed back to the calling client.

For a PowerShell serialized request it looks like this:

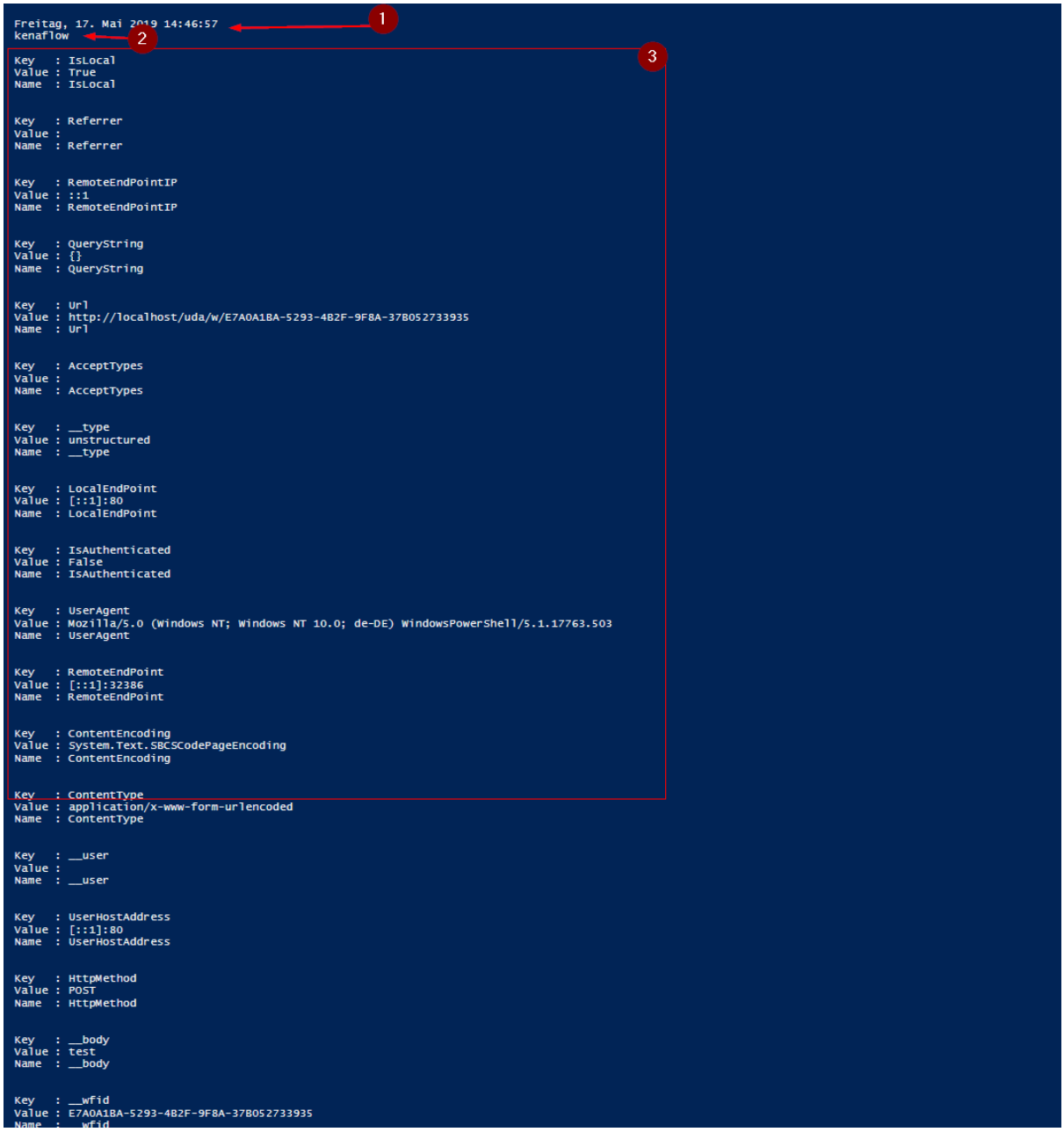
```
    $result = (Invoke-WebRequest "http://localhost:80/uda/w/E7A0A1BA-5293-4B2F-
↪9F8A-37B052733935" -Body "test" -Method "POST")
```

See the "w/" in it?

I deserialize the response body with this and look at the two result objects

```
    $psResult = [System.Management.Automation.PSSerializer]::Deserialize([System.
↪text.encoding]::utf8.getstring($result.Content))
    $psResult.CurrentDate
    $psResult.Test
    $psResult.Evt
```

Thats the result:



(1) is `$psResult.CurrentDate`(2) is `$psResult.Test`(3) is `$psResult.Evt` - the actual event object that is passed back to the calling client. Here it is deserialzed not as an PSObject but as Hashtable. (JSON/the serializer/the deserializer does not know that it is a Hashtable instead of an object.)

You can use this functionality to communicate with a workflow on a website through AJAX requests. If you want to test such an request from JavaScript you could create a simple HTML page with an jQuery based AJAX request:

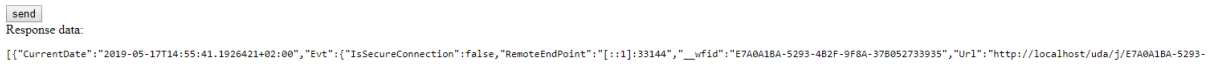
```
    <html>
    <script src="jquery.js"></script>
    <script type="text/javascript">
    function send(){
            $.post('http://localhost:80/uda/j/E7A0A1BA-5293-4B2F-9F8A-37B052733935/
↪',
                JSON.stringify({ "test":"kenaflow is cool!"})
        ).done(function(data, State) {
                    $('#response').html('Response data: ');
                    $('#data').html(data);
            });
```

(continues on next page)

(continued from previous page)

```
    }
    </script>
    <body>
    <input type="button" onclick="javascript:send();return false" value="send"/>
    <div id="response"></div>
    <pre id="data"></pre>
    </body>
    </html>
```

This is the result in the browser:



When you use `/b` to return binary data the first value in the `Accept` http request header will be used as `Content Type` in the http response header.

The response content type can also be specified by a query string parameter called `__as` with the content type as value. (See example below.) If this is available it is used instead of the `Accept` http request header.

The workflow script should return one or more byte arrays. If there are more than 1 the will be concatinated. Also possible is Base64 a return value. It is concatinated with the other results to a byte array. - Returned strings from the workflow script are tested to be Base64 and get concatinated if they are. If they are not Base64 they are converted to `Byte` array using encoding UTF-8. After that they are concatinated to the `Byte` array.

With that it would be possible to create a dynamic PDF in the browser:

`https://kenaflow.local/<instance-id>/uda/b/<powershell-workflow-id>/?__as=application/pdf`

The workflow script could look like this:

`_wfconfig.ps1`

```
    @{
        Version = "4.0";
        Enabled = $true;
        Type = "POWERSHELL";
        Name = "get-pdf";
        Id = "7c24e165-21c2-418e-ab3e-c6702837b533";
        TBE = -1;
        ConfigListDefaults = @{
            "kenaflow" = "cool"
        };
        AlertAddresses = @();
        MailFrom = "";
        MailReply = "";
        Script = "script.ps1";
        RER = $true;
    }

    if(!$kenaflow){import-module "D:\kenaflow\engine\kenaflow.runtime.dll";Test-
↪KFConfig;exit}
```

`script.ps1`

```
    param($wf, $eventData, $config)
    if($wf-eq$null){import-module "D:\kenaflow\engine\kenaflow.runtime.dll";Invoke-
↪Kenaflow;exit}

    if($eventData) {
```

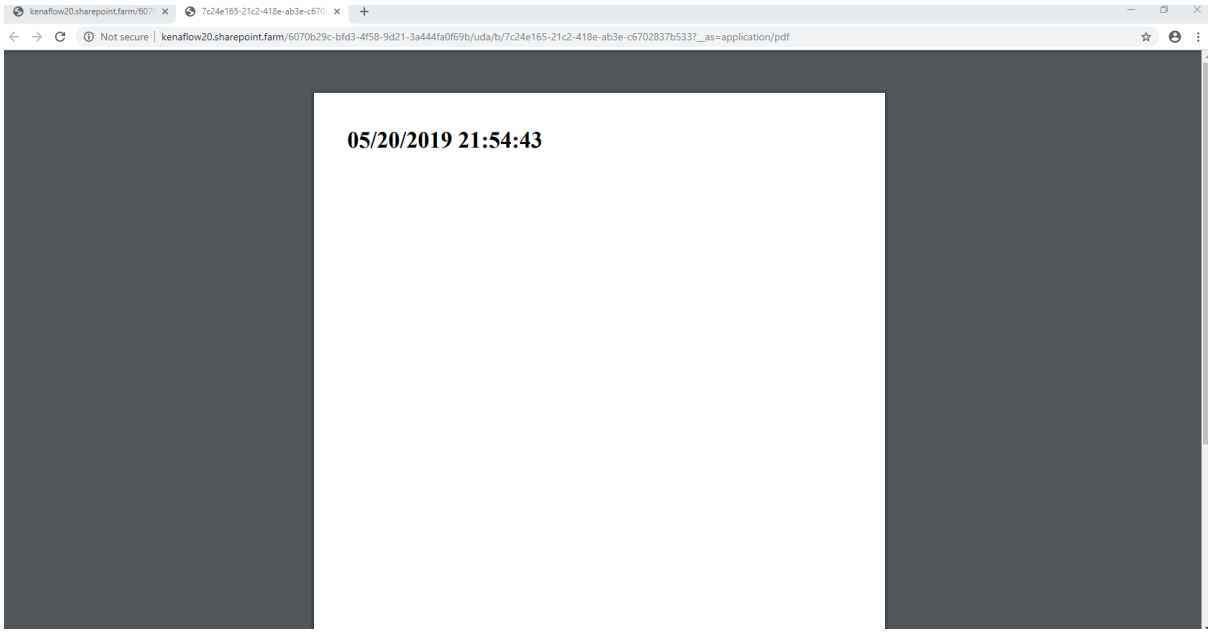
(continues on next page)

(continued from previous page)

```
            return (New-KFPDF -html "<html><body><h1>$([Datetime]::Now)</h1></body>
↪</html>")
    }
```

test.bat

```
    "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" "http://
↪kenaflow20.sharepoint.farm/6070b29c-bfd3-4f58-9d21-3a444fa0f69b/uda/b/7c24e165-
↪21c2-418e-ab3e-c6702837b533?__as=application/pdf"
```



11.11.8 Link Remote Events

This type of remote event is described here: *Links*

11.11.9 Behaviour in Case of Errors

If an error / exception occurs during remote event execution the remote event will be postponed.

The amount of seconds is configurable in `_wfconfig.ps1`. If there is no value configured the setting from global configuration will be used. There is a default of 5 seconds.

Postponing will happen again and again if the remote evens fails.

There is a configurable maximum lifetime for a remote event. The setting is taken from `_wfconfig.ps1`. If there is no value for that the global configuration setting will be used. The default there is 120 seconds.

11.12 Error Handling

11.12.1 Error Handling Basics

Error handling in scripts is a very importent task for a *kenaflow* workflow developer.

!! Nobody's perfect!

Errors can happen - as you know - by making things wrong or by having errors in the software (*kenaflow* => could happen, sorry) or in the infrastructure (SharePoint, Exchange, mail server, network infrastructure, . . .)

Normally, all these errors cause exceptions to be thrown in the script.

kenaflow takes exceptions and reports them to the developer. *kenaflow* also sends emails in case of error, if this is configured in `_wfconfig.ps1` or in the global configuration.

There are also possibilities to handle errors before they reach the engine level, i.e. to handle the exceptions within the workflow.

Basically, all mechanisms of PowerShell that every PowerShell developer knows can be used.

```
try {

        throw "an exception"
} catch {
        Write-KFLog "Handeld error: $($_.Message)"
}
```

This error is handlet (catched) and not send to *kenaflow*. The engine will not handle this error because it never get informed about this error.

All cmdlets of *kenaflow* (that are not part of the embedded "SharePoint Patterns and Practices PowerShell" module) habe the default switch parameter `-ThrowErrors` with a default value of `$true`. It can be used to "hide" exceptions.

If specified like this:

```
Get-KFUserInfo "this is not a valid user" -ThrowErrors:$false
```

. . . the exception will not be thrown by the script and not handled by *kenaflow*. It is also not catch-able with `try...catch...`.

Therefore you can use *Get-KFLastResult* to get the result of the last *kenaflow* cmdlet. If an exception occured you will get this exception.

```
Get-KFUserInfo "this is not a valid user" -ThrowErrors:$false
if( (Get-KFLastResult) -is [System.Exception] ) {
        Write-KFLog "Something went wrong".
}
```

Instead of specifying `-ThrowErrors` on each cmdlet you can use the cmdlet *`Set-KFThrowErrorsPreference`* to set the default.

You can use to cmdlet *`Get-KFThrowErrorsPreference`* to get the current default.

The workflows default is configured in `_wfconfig.ps1`. If nothing is set exceptions are thrown by default!

11.12.2 Error Handling Scripts

In addition to these options, there is also the option to define an "error handling script" for a workflow.

The script is specified in `_wfconfig.ps1`.

As described above if an error / exception occurs the workflow script is aborted. Normally *kenaflow* handles the error. The default handling is to stop the workflow and inform the developers - if configured - via email.

If an error handling script is configured it is used to process the exception.

The default error handling script is named `errorHandling.ps1` and has the following default content. The first line for the parameters differes for the different workfow types. Here is the line for **SharePoint list workflows** and **SharePoint state machine workflows**.

```
    param($wf, $config, $exception, $item, $eventData)

    if($null-eq$wf){write-error "You need to run a workflow script and force an␣
↪error.";exit;}


    #Return one of these values:
    # 0 or "ThrowException" or [KenaflowErrorHandlingResults]::ThrowException
    # 1 or "Okay" or [KenaflowErrorHandlingResults]::Okay
    # 2 or "StopItemWithSuccess" or␣
↪[KenaflowErrorHandlingResults]::StopItemWithSuccess
    # 3 or "StopStateWithSuccess" or␣
↪[KenaflowErrorHandlingResults]::StopStateWithSuccess
    # 4 or "StopWorkflowWithSuccess" or␣
↪[KenaflowErrorHandlingResults]::StopWorkflowWithSuccess
    # 5 or "StopItemWithFailure" or␣
↪[KenaflowErrorHandlingResults]::StopItemWithFailure
    # 6 or "StopStateWithFailure" or␣
↪[KenaflowErrorHandlingResults]::StopStateWithFailure
    # 7 or "StopWorkflowWithFailure" or␣
↪[KenaflowErrorHandlingResults]::StopWorkflowWithFailure
    #
    # "ThrowException" is default in case nothing is send back
    #
    # "Okay" indicated that the error handling script has handled the exception␣
↪and the engine must not take care.
    #
    # You cannot send other data back to _kenaflow_!

    return 0
```

Here is the description of the script parameters:

- `$wf` denotes the workflow configuration, which contains all the settings defined in the workflow configuration file `_wfconfig.ps1`.
- `$config` is a PowerShell hashtable that contains the additional configuration stored in SharePoint. See chapter *Workflow Lists*.
- `$exception` contains the exception that caused the abortion of the workflow script.
- `$item` contains the current SharePoint List item (for **State Machine Workflows** or **List Workflows** or **Email Workflows**). It is `$null` in case of executing exceptions from the pre-query script (for **State Machine Workflows** or **List Workflows**) and it DOES NOT EXIST AT ALL for **SharePoint Site Workflows** and **PowerShell Workflows**
- `$eventData` contains the current remote event data. Or it is `$null` if there is no event data. For **Email Workflows** this parameter does not exist!

For **SharePoint Site Workflows** and **PowerShell Workflows** the first line looks like this:

```
    param($wf, $config, $exception, $eventData)
```

For **Email Workflows** the first line looks like this:

```
    param($wf, $config, $exception, $item)
```

!! If an error occured inside the error handling script an exception is passed to the workflow engine. There is no "error handling script for the error handling script". (Yes, we got this question once!)

In case of an *Email* workflow or an SharePoint *List* workflow `StopStateWithSuccess` behaves as `StopWorkflowWithSuccess` and `StopStateWithFailure` behaves as `StopWorkflowWithFailure`.

In case of an *PowerShell* workflow `StopItemWithSuccess` and `StopStateWithSuccess` behaves as `StopWorkflowWithSuccess` and `StopItemWithFailure` and `StopStateWithFailure` behaves as `StopWorkflowWithFailure`.

In case of errors inside the pre-query script `StopItemWithSuccess` and `StopStateWithSuccess` behaves as `StopWorkflowWithSuccess` and `StopItemWithFailure` and `StopStateWithFailure` behaves as `StopWorkflowWithFailure`.

11.12.2.1 Failure / Success

In case of Email workflows a "Failure" means that the email item is not marked as "read" (IMAP or EXCHANGE) or deleted (POP3).

For all workflow types "Failure" means, that the next run of the workflow is not controlled by "time between execution". The workflow should run again as soon as possible.

11.13 Persisting Data

There are many ways to save data from a workflow and reload it for a subsequent workflow run.

11.13.1 File System

The workflow developer can save data himself in files in the workflow folder. It is up to him/her to do this or to choose this path.

11.13.2 Database

Any database system accessible from within PowerShell can be used by the workflow developer.

11.13.3 *kenaflow* Item Context Persistence

The following option is only available in SharePoint *list* workflows and *state machine* workflows.

kenaflow offers an easy way to save data in the context of the processed SharePoint list items and reload it later.

For this purpose, the workflow *data list* is used. This is a simple custom list, creates by *kenaflow* using program switch `--createdatalist`. The data list must be named in the `_wfconfig.ps1` workflow config file.

The data list contains a key column and two value columns, one for CliXML format and one for JSON format.

When using cmdlet `Set-KFItemData` the workflow developer can serialize an object to the data list using a given key. The key is augmented with the unique list item id.

When loading the data later with *Get-KFItemData* the same key must be used. (Only the CliXML column is read by *kenaflow* during reading!) It is again augmented with the unique list item id. The object is read from the data list and gets deserialized.

The workflow should clean up the data list using *Remove-KFItemData*. Otherwise the data is never removed from the data list!

11.13.4 *kenaflow* Data Persistence

The following option is only available in SharePoint workflows (*list*, *state machine* and *site* workflows).

Data can be serialized to the workflow *data list* as CliCML and JSON with a key.

It can be used to exchange data between multiple workflow (if the share the same data list) or between different processed list items.

When using cmdlet *Set-KFData* the workflow developer can serialize an object to the data list using a given key. The key is used "as is".

When loading the data later with *Get-KFData* the same key must be used. (Only the CliXML column is read by *kenaflow* during reading!) (The key is used "as is".) The object is read from the data list and gets deserialized.

The workflow should clean up the data list using *Remove-KFData*. Otherwise the data is never removed from the data list!

11.13.5 *kenaflow* SQLite Persistence

The following option is available in all workflow types.

It uses an SQLite database normally located in the program data folder of *kenaflow*. The location can be configured in the global configuration *Section "Basic"*, key "database".

By using a key data objects can be written to the database using *Set-KFDBData*, read from the database with *Get-KFDBData* and removed from the database with *Remove-KFDBData*.

The data is stored in CliXML serialized form in the database.

The data can be written/read/removed in context of the workflow which means that the workflow id is used additionally to the given key.

In case of SharePoint *list* and *state machine* workflows also the unique list item id can additionally be used.

11.14 Trigger

Sometimes it is necessary to start a workflow from another one.

Therefore we have prepared cmdlet *Invoke-KFWorkflow*.

!! You can only trigger SharePoint workflows (*list*, *state machine*, *site*) and PowerShell workflows.

There you name the workflow by its id and optionally specify a list item id.

The called workflow is started. The trigger is passed as event to the workflow script in parameter `$eventData`.

The `$eventData` object will be a PowerShell Hashtable with this entries:

- `__sourcewfid`
- `__wfid`
- `__itemid`
- `__sourcewfitemid`
- `__data`

Data cann be passed to the called workflow with the `-Data` parameter of `Invoke-KFWorkflow`.

The data that is transfered to the called workflow must be serializable. It's not possible to transfer complex object such as System.IO.Stream.

11.14.1 Return Data

It is possible to send data back to the caller.

Here are two workflows.

The "Caller" config (`_wfconfig.ps1`):

```
    @{
            Version = "4.0";
            Ignore = $true;
            Type = "SHAREPOINT";
            SubType = "SITE";
            Name = "trigger-sender";
            Platform = "sp2013";
            Id = "52989b84-43c5-4db8-ae10-4f04dcb03e6c";
            TBE = 60;
            CustomLibrary = "";
            spUser = "sharepoint\kenaflow";
            spPwd = 'kr@ftwerk';
            Web = "https://intranet13.sharepoint.farm";
            Script = "script.ps1";
    }

    if(!$kenaflow){import-module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Test-KFConfig;exit}
```

The workflow script `script.ps1`

```
    param($wf, $web, $config, $eventData)

    if($wf-eq$null){import-module "C:\Source\wfengine\bin1\Debug\kenaflow.runtime.
↪dll";
      Invoke-Kenaflow -Breakpoint;exit}

    $result =
      (Invoke-KFWorkflow \`
        -WorkflowId "8808aecd-f7e5-444c-a625-0d1506c2e99c" \`
        -WithAnswer -AnswerTimeout 10000)

    Write-KFLog "kenaflow is $($result.kenaflow)!"
```

The called workflows config (`_wfconfig.ps1`):

```
    @{
            Version = "4.0";
            Ignore = $false;
            Type = "POWERSHELL";
            Name = "trigger-receiver";
            Id = "8808aecd-f7e5-444c-a625-0d1506c2e99c";
            TBE = -1;
            RER = $true;
            Script = "script.ps1";
    }
```

The called workflows script (`script.ps1`):

```
    param($wf, $web, $item, $config, $eventData)

    if($wf-eq$null){import-module "C:\source\wfengine\bin1\Debug\kenaflow.runtime.
↪dll";
      Invoke-Kenaflow -debug -verbose;exit}

    return @{kenaflow="cool"}
```

The first workflow will output "kenaflow is cool!" to the log. "cool" comes from the called workflow as return value.

!! The return data must be serializable!

11.15 The kenaflow Field

kenaflow for SharePoint processes list items with *list* and *state machine* workflows. Only to this workflow types belongs this article.

11.15.1 General Information

The *kenaflow* field stores data for helping *kenaflow* to process the right list items. Mainly it contains the date and time of the last workflow run.

Because multiple workflows can operate on the same list it is necessary to store this workflow specific data with the workflow ID as a key.

Multiple *kenaflow* instances or servers could operate on the same SharePoint list. This is not recommanded because in this way it is more complicated to lock list items during processing. (All *kenaflow* instances should use the same "lock folder" than.) - To be able to read and write the *kenaflow* field from different *kenaflow* instances this field is **not** encrypted. But it does not contain sensitive data!

11.15.2 What is it for?

The *kenaflow* field in the first place supports the execution of so called "last modified" workflows.

"LAST MODIFIED" workflows or workflow states only operate on items that where modified since the last run of the workflow.

BUT *kenaflow* tries to keep the information about the last editor and the last modification time so that the work of the engine is as transparent as possible for the users. After completion of the work, the fields "Editor" and "Modified" are set to the values *before* processing by *kenaflow*. However, this makes it difficult to execute the "LAST MODIFIED" workflow types: During the first run - with modified item - the item is changed by the workflow (or not, depending on the workflow script and intention of the workflow developer) and finally the date of the last modification is set back to the value *before* processing. At the second run of the engine this item is found *again*, because the "Modified" date is set to the same value as at the first run.

To solve this problem we have introduced the "kenaflow field". It is a hidden field that is added to every SharEPoint list and library where *kenaflow* works.

From a technical perspective this field is defined as follows:

```
    <Field
      Type='Note'
      ID='06253ACC-EB61-47B6-9D8E-521550E29F5F'
      Hidden='TRUE'
      UnlimitedLengthInDocumentLibrary='TRUE'
      DisplayName='kenaflow'
      StaticName='kenaflow'
      Name='kenaflow'/>
```

It gets added to "all content types" of the list or library!

11.15.3 Change Hash

Since *kenaflow* version 4.0 the *kenaflow* field does contain the change hashes for SharePoint list items when you use the property `ItemChangeHashFields` in the workflow. This means that the workflow does only process list items with changes in the specified field(s). After processing the item the change hash is calculated based on the current item field values. If they are not changed before the next workflow run the item will not be processed.

11.16 Storage

"Storage" means that data can be stored to and retrieved from disk.

The data can be encrypted on disk!

The workflow developer is able to add a passphrase for encryption. This is *added* to the *kenaflow* passphrase. This means that the workflow developer only knows a part of the whole passphrase!

It is written to a folder on the file system or network share. The folder is configured in global configuration *section "Storage"*.

11.16.1 Permanent Storage

Permanent storage offers the ability to store data on the storage that is available on further workflow runs.

The data is stored in a subfolder of the configured storage folder. The subfolder is the workflow id or the "empty GUID" (`00000000-0000-0000-0000-000000000000`).

The file name of the stored data is defined by the workflow developer: the given `key` is converted into a valid filename which means that invalid characters are escaped.

Encrypted objects file names have an file name extension of `.e`. Not encrypted object file names have no extension.

The following 3 cmdlets can be used to access the permanent storage.

- *`Set-KFStorageData`*
- *`Get-KFStorageData`*
- *`Remove-KFStorageData`*

11.16.2 Session Storage

Session storage offers the ability to store data on the storage that is removed after the workflow execution.

For session storage a subfolder below the storage folder is created. The name of the subfolder starts with `s-` followed by the machine name and the process ID.

Below this session storage folder the data is stored in a subfolder named with the workflow id or the "empty GUID" (`00000000-0000-0000-0000-000000000000`).

The file name of the stored data is defined by the workflow developer: the given `key` is converted into a valid filename which means that invalid characters are escaped.

Encrypted objects file names have an file name extension of `.e`. Not encrypted object file names have no extension.

The following 3 cmdlets can be used to access the session storage.

- *`Get-KFSessionStorageData`*
- *`Set-KFSessionStorageData`*
- *`Remove-KFSessionStorageData`*

11.17 Security Hardening

kenaflow is a software that has been designed and developed for use in the large customer environment. We call it "Enterprise ready".

For a "security hardening" it is essential that the rights of users and administrators are limited to the respective minimum, so that each person / role only has exactly the rights that are necessary for the execution of the function / role.

There are the following levels in *kenaflow* that must be considered for this:

1. access to SharePoint
2. access to workflow scripts / workflow configuration
3. instance security

11.17.1 Access to SharePoint

For the execution of actions *in* SharePoint, the respective workflow must *access* SharePoint. This means that data is read and possibly written. Access to SharePoint must be configured accordingly.

There are two possibilities:

1. an execution account for the *kenaflow* instance
2. an access account for SharePoint.

11.17.1.1 Execution Account

An execution account can only be used with SharePoint On-Premises.

In this case the software *kenaflow* is executed in the context of a function user ("Service Account").

In SharePoint, this account receives exactly the access rights that are necessary to read and/or write data via the workflow.

The assignment of rights is carried out by the workflow developer or by a SharePoint administrator.

For example, if a workflow has to read a SharePoint list, it is NOT necessary that the execution account is allowed to read all lists of the SharePoint page.

The execution account is a Active Directory Domain user account with no special permissions.

For security hardening this account should be NO local administrator on the *kenaflow* server. But as it is used to run *kenaflow* as scheduled task it need the permission "Logon as a batch job" in the Windows Local Security Policy.

11.17.1.2 Access Account

It is also possible to specify an access account for a workflow. The software itself can then be executed on the *kenaflow* server with an account that has no rights in SharePoint.

The execution account is then an account with the appropriate rights in SharePoint.

This could be an Active Directory Domain account account for SharePoint On-Premises. Or an Azure Active Directory account that is registered in SharePoint Online.

In case of using an access account the *kenaflow* instance can run with an local machine account instead of an Active Directory Domain account.

Also an Windows Managed Account can be used. No user and no administrator will have the password for such an account.

"App Credentials" can also be used. This works both in SharePoint On-Premises and SharePoint Online. Read this: *App ID & App Secret*

The credentials for the access account are stored in the workflow configuration file `_wfconfig.ps1`. To **encrypt** the setting the command line switches `--setusername` or `--appid` are used. The credentials get strongly encrypted with an key that is specified in the global configuration `encryptionPassphrase` in *section "AfterSetup"*. The global configuration itself is encrypted using a *machine specific* key that is only known to Windows OS. In theory every user with priviledge to access the *kenaflow* server could decrypt the global configuration. Therefore reverse engeneering of the *kenaflow* software is required besides the access to the server.

11.17.2 Access to Workflow Scripts / Workflow Configuration

Workflows in *kenaflow* consist of a series of PowerShell scripts. These are located in the file system.

Each workflow is located in a separate directory below the "Workflow Root" directory. Additional layers can be added below "Workflow Root", but they may only contain additional folders and not workflows themselves.

With Windows file system permissions, access to workflow scripts can be prevented for workflow developers. This can be used if workflow developers are logged on to the *kenaflow* server via RDP for workflow development.

Workflow developers can develop workflows on their own machines and then have them transferred to the productive *kenaflow* server via automated procedures - e.g. "Continuos Integration" - so that direct access to the server by the developer is not necessary. This is the recommended procedure in a "security hardening" scenario:

1. Developers use their own machines for workflow development. They have their own *kenaflow* installation, Developer Edition. (See: Editions](#1c26a619-cf76-44b9-9dcf-4214b1e12b6b))
2. Developers use their own account for workflow execution in debug mode or runtime mode. The account must have appropriate permissions in the SharePoint development environment that is identical to the production environment (in terms of SharePoint artifacts like content types and column/field names).
3. After development the developers checkin the code in a source control system like GIT.
4. An automatic process in a continuos integration system like JENKINS registeres the code change and triggers a job that is allowed to copy the workflow code from the source control to the productive *kenaflow* instance.
5. There the workflow is executed with the execution account of *kenaflow* that has the appropriate permissions in SharePoint production environment.

11.17.3 Instance Security

During setup of *kenaflow* the required execution account must be set to run *kenaflow* automatically as Windows scheduled task.

`--installschedtask`

Using the command line switch `--securing` the permissions of the execution account are set on the appropriate file system locations.

For details please read this article: *System Changes*

With command line switch `--confighttp` the HTTP(S) endpoints of *kenaflow* are configured.

With command line switch `--updatenetsecurity` the HTTP(S) endpoint security can be modified after the setup, e.g. if the execution account was changed.

Workflow developers on the *kenaflow* server must have read permissions on the *kenaflow* binaries folder BUT NO WRITE PERMISSIONS there!

Workflow developers on the *kenaflow* server must have read & write permissions on the *kenaflow* data folder. This can be a network share.

For security hardening it is recommanded to use dedicated development machines for developers instead of developing workflows on the production server.

11.18 Pre-Defined Variables

Starting with *kenaflow* 4.0.22 the engine injects several pre-defined variables into the PowerShell script during execution.

The content of the variables depends on the current workflow, execution state, etc.

Depending on the script context there are different variables available.

All pre-defined variables start with prefix `KF`.

11.18.1 Workflow Configurtation Parsing

The workflow configuration file `_wfconfig.ps1` is a PowerShell script that returns a single Hashtable object containing the workflow configuration. This file is parsed using PowerShell. Therefore the code of `_wfconfig.ps1` gets executed.

The following variables are injected:

- `$KFInstanceName`
- `$KFInstanceId`
- `$KFWorkflowConfigFileInfo` => Object of type `System.IO.FileInfo`

11.18.2 `PreWorkflowExecution` Script Block

When debugging a workflow script the cmdlet `Invoke-Kenaflow` is used to initialize the runtime environment.

The cmdlet accepts a script block with parameter `-PreWorkflowExecurion`. This script block is executed before the workflow script is executed.

Inside the scriptblock the following variables are injected:

- `$KFInstanceName`
- `$KFInstanceId`
- `$KFWorkflow`
- `$KFConfig`
- `$KFLastRun`
- `$KFCurrentStart`
- `$KFLastRunDT`
- `$KFCurrentStartDT`
- `$KFLastRunUtc`
- `$KFCurrentStartUtc`
- `$KFLastRunUtcDT`
- `$KFCurrentStartUtcDT`

For different workflow types there are additional variables available.

11.18.2.1 Additional Variables in EMail-IMAP Workflows

- `$KFIMAPServerObj`

11.18.2.2 Additional Variables in EMail-POP3 Workflows

- `$KFPOP3ServerObj`

11.18.2.3 Additional Variables in EMail-Exchange Workflows

- `$KFExchangeWSServerObj`

11.18.2.4 Additional Variables in SharePoint Enabled EMail Workflows

- `$KFCurrentWeb`
- `$KFContext`
- `$KFSharePointContext`

11.18.2.5 Additional in Storage Workflows

- `$KFEventData`

11.18.2.6 Additional Variables in in (all) SharePoint Workflows

- `$KFCurrentWeb`
- `$KFContext`
- `$KFSharePointContext`
- `$KFEventData`

11.18.2.7 Additional in SharePoint State Machine Workflows

- `$KFWorkflowStates` => An object of type `Dictionary<string,int>` that contains state names and ids.

11.18.2.8 Additional Variables in PowerShell Workflows and Custom Workflows

currently none

11.18.3 ErrorHandling-Script

In a workflow an "Error Handling Script" can be specified in `_wfconfig.ps1`.

The scripts is called with some information about the error (`Exception`) and the following pre-defined variables:

- `$KFException`
- `$KFInstanceName`
- `$KFInstanceId`
- `$KFWorkflow`
- `$KFConfig`

The following variables are optionally injected, depending on the context.

- `$KFCurrentItem`
- `$KFEventData`
- `$KFCurrentWeb`
- `$KFCurrentList`

11.18.4 PreProcessing Script and PostProcessing Script

Every workflow can have a "PreProcessing" script and/or a "PostProcessing" script.

The following variables are injected during runtime:

- `$KFInstanceName`
- `$KFInstanceId`
- `$KFWorkflow`
- `$KFEventData`

11.18.5 (All) SharePoint Workflows

- `$KFInstanceName`
- `$KFInstanceId`
- `$KFWorkflow`
- `$KFContext`
- `$KFSharePointContext`
- `$KFCurrentWeb`
- `$KFEventData`
- `$KFConfig`
- `$KFLastRun`
- `$KFCurrentStart`
- `$KFLastRunDT`
- `$KFCurrentStartDT`
- `$KFLastRunUtc`
- `$KFCurrentStartUtc`
- `$KFLastRunUtcDT`
- `$KFCurrentStartUtcDT`
- `$KFInVersionConflict`

11.18.5.1 Additional in SharePoint List Workflows and State Machine Workflows

- `$KFItem`
- `$KFCurrentList`
- `$KFLastEditor`
- `$KFLastModified`
- `$KFLastItemModifiedFlag`
- `$KFLastExecutionType`

11.18.5.2 Additional Variables in SharePoint State Machine Workflows

- `$KFCurrentWorkflowState`

11.18.6 Pre Query Script of SharePoint List and State Machine Workflows

SharePoint List workflows and SharePoint State Machine workflows can have a script "PreQuery" that is spcified in `_wfconfig.ps1` and is executed just before the CAML query is send to SharePoint to retrieve the items.

The following variables are injected into the Pre Query Script:

- `$KFInstanceName`
- `$KFInstanceId`
- `$KFWorkflow`
- `$KFContext`
- `$KFSharePointContext`
- `$KFCurrentWeb`
- `$KFCurrentList`
- `$KFEventData`
- `$KFConfig`
- `$KFLastRun`
- `$KFCurrentStart`
- `$KFLastRunDT`
- `$KFCurrentStartDT`
- `$KFLastRunUtc`
- `$KFCurrentStartUtc`
- `$KFLastRunUtcDT`
- `$KFCurrentStartUtcDT`
- `$KFInVersionConflict`

11.18.7 Custom Workflow

Custom workflows can be defined using *kenaflow* Plugins.

The following variables are injected into the script execution:

- `$KFPlugin` => A reference to the *kenaflow* Plugin used for this workflow
- `$KFConnection` => Depending on the *kenaflow* Plugin
- `$KFEventData`

11.18.8 EMail Workflow

The following variables are available during execution:

- `$KFInstanceName`
- `$KFInstanceId`
- `$KFItem`
- `$KFWorkflow`
- `$KFConfig`
- `$KFLastRun`
- `$KFCurrentStart`
- `$KFLastRunDT`
- `$KFCurrentStartDT`
- `$KFLastRunUtc`
- `$KFCurrentStartUtc`
- `$KFLastRunUtcDT`
- `$KFCurrentStartUtcDT`
- `$KFInVersionConflict`

11.18.8.1 Additional Variables in SharePoint Enabled EMail Workflows

- `$KFContext`
- `$KFSharePointContext`
- `$KFCurrentWeb`
- `$KFInVersionConflict`

CHAPTER

TWELVE

ADVANCED CONFIGURATION

In this section, we explain advanced configuration options to be able to *kenaflow* more precisely to the concrete needs.

12.1 Storage

Details about "Storage"

12.2 Execution Slots

Execution slots are an advanced feature in kenaflow.

To seperate different workflows from others you can use this feature.

Details about "Execution Slots"

12.3 Workflow Priority

In *kenaflow* workflows are executed in parallel. All workflows are queued for execution. To control this queue for important workflows you can use *"Workflow Priority"*

12.4 Storage

kenaflow stores data for workflow execution in files. The operating system controls very efficiently which of these files are stored on the hard disk and which only remain in the main memory.

If *kenaflow* is to be scaled over several servers, then it is necessary that the cooperating *kenaflow* instances share different data directories.

In this case this is solved by network shares.

For performance reasons it might be useful to put certain kenaflow control files into a "RAM disk".

We recommend ImDisk Toolkit: https://sourceforge.net/projects/imdisk-toolkit/

!! We do not assume any guarantees and/or liability for the use of third-party tools.

12.5 Execution Slots

Execution slots are an advanced feature in kenaflow.

They can be used to execute workflows on different levels or "execution lanes" so that they work on distributed execution venues.

Workflows in kenaflow are executed in parallel. For this purpose sub processes are executed. Each sub process executes one workflow. All workflows are in a common queue. The maximum number of parallel executions is defined in the global configuration.

Please read this article for *the details*.

Workflows usually have a "TBE" time: Time Between Execution. This specifies how many seconds must elapse between two starts of the workflow before it is executed again. The specification of the "TBE" is a "minimum" specification. There is no guarantee that the workflow will restart after this time, since other workflows may be scheduled in the queue beforehand.

If it is now necessary for certain workflows to have a highly probable execution lane in the specified time, then execution slots can be used.

An analogy: Imagine a 5-lane highway. Each highway lane is a *kenaflow* execution lane. Slow workflows can prevent the fast ones from getting past them when the slow ones are allowed to occupy all lanes. Execution Slots can be used to specify that certain workflows are to be defined for certain lanes. Slot "0" (zero) accepts all non-configured workflows.

Any slots can be defined via the global configuration. This is configured in the "slots" setting in the *"Main Mode"* section. A list of slots is created here by specifying how many execution venues are provided per slot.

For example, if you have this settings in global configuration:

```
        <entry key="parallelWFExecutors" type="System.Int32" default="10"></entry>
        <entry key="slots" type="System.String" comment="semicolon separated list␣
↪of integer values" default="">2;3</entry>
```

. . . than you have configured 10 execution lanes ("parallelWFExecutors"). And you have configured **3** slots: Slot #1 with 2 execution laness, Slot #2 with 3 execution lanes - and Slot #0 (default slot) with the rest of 4 execution lanes (10 - 2 - 3 = 7).

In the workflow configuration (***missing***) you can specify on which slot the workflow should run. All workflows that are configured for a certain slot will share all execution lanes that are assigned to the slot.

12.6 Workflow Priority

In *kenaflow* workflows are executed in parallel. You configure how much can run this way at the same time. It depends on the capabilities of the server and of the performance and capabilities of your SharePoint farm because the farm has to handle all the client requests made by *kenaflow*.

All workflows are queued for execution. Most workflows have a setting called "TBE" (=Time Between Execution) which specifies the amount of time between to starts of the same workflow.

When a workflow should run again because of its TBE and the last start time it is not guaranteed that it is executed immediatemy because other workflows could run at that time and it's possible that others are also waiting in the queue with an higher position in it.

BUT you can use "Workflow Priority" to move workflows up in the queue.

The priority is a integer value in the `_wfconfig.ps1` workflow config file.

Each workflow has priority 0 (zero) by default.

You can specify any integer value you like.

The priority engine will figure out what to do. If an execution lane (that's how we call one parallel execution opportunity) is free for the next workflow the priority engine compares all waiting workflows with their priority settings, TBE, last start time and other factores. It decides what workflows will run next on the free execution lane.

There is a way to simulate the priorty engine to see how workflow execution is affected by it.

(Will be written later.)

12.7 Advanced Workflow Configuration

The workflow config `_wfconfig.ps1` itself is a PowerShell script.

It is executed:

- if it was changed since the last workflow run
- if *kenaflow* was updated

In this case the engines executed the config file and serializes the result in an encrypted fashion on disk in the `_wfdata` subfolder of the workflow folder.

Because the workflow is a script you are able to do you own stuff. But you need to keep in mind that it is **not executed on every workflow run** for performance reasons.

You can tell the engine **not** to serialize the config file. In this case it **will** execute on every workflow run!

To do so you can specify property `NoSerializing = $true` in the `_wfconfig.ps1`.

But it's not always necessary to prevent serialization. Most of the time it will be sufficient to execute it once.

You can use own code to configure the workflow.

For example if you want to configure the SharePoint web and list the workflow is working at depending of the workflows folder:

This is a pseudo-code `_wfconfig.ps1`:

```
    $path = Split-Path $MyInvocation.MyCommand.Path
    $list = ""
    if( $path -like "c:\workflows-production\*") {
        $list = "Custom List Production"
            $web = "Production Web"
    } else {
        $list = "Custom List Dev"
            $web = "Dev Web"
    }

    @{
        #... other settings ...

            Web = $web;
            List = $list;

            #... other settings ...
    }

    if(!$kenaflow){Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Test-KFConfig;exit}
```

CHAPTER

THIRTEEN

AGENTS

kenaflow supports it's own "agents" for remote powerShell execution.

This will be described here.

CHAPTER

FOURTEEN

PLUGINS

Currently there are two architecture versions for plugins.

14.1 Plugin Architecture Version 1

This architecture version of *kenaflow* plugins is intended to allow plugin developers to create and integrate *custom PowerShell cmdlets* that are mounted in *kenaflow* workflows.

There is a public demo plugin on GitHub:

https://github.com/ikarstein/kenaflow.demoplugin.v1

Please read article *Plugin Development*

14.2 Plugin Architecture Version 2

This architecture version of *kenaflow* plugins is intended to allow plugin developers to create *custom workflow types*. It can also contain *custom PowerShell cmdlets*.

There is a public demo plugin on GitHub:

https://github.com/ikarstein/kenaflow.demoplugin.v2

Please read article *Plugin Development*

14.3 Plugin Development

There are public demo plugins on GitHub:

- https://github.com/ikarstein/kenaflow.demoplugin.v1
- https://github.com/ikarstein/kenaflow.demoplugin.v2

14.3.1 Setup

They show what to do to develop *kenaflow* plugins.

To set this up just clone the repository

```
git clone https://github.com/ikarstein/kenaflow.demoplugin.v1
```

or

```
git clone https://github.com/ikarstein/kenaflow.demoplugin.v2
```

Now you need run PowerShell script `_patch.ps1`

```
. .\_patch.ps1 "<path-to-kenaflow>"
```

By default *kenaflow* is installed in `C:\Program Files\kenaflow`. Therefore run:

```
. .\_patch.ps1 "C:\Program Files\kenaflow"
```

The project has an assembly reference to `kenaflow.lib.dll`. Furthermore it has a NuGet package reference to `Microsoft.PowerShell.5.ReferenceAssemblies` (Version 1.1.0).

**Both assembly references have a setting `Private` (In Visual Studio Solution Explorer it's calles 'Copy Local') with value `False` in the project file.

```
        <Reference Include="kenaflow.lib">
          <HintPath>c:\program files\kenaflow\kenaflow.lib.dll</HintPath>
          <Private>False</Private>
        </Reference>
        <Reference Include="System.Management.Automation, Version=3.0.0.0,␣
↪Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL">
          <HintPath>packages\Microsoft.PowerShell.5.ReferenceAssemblies.1.1.0\lib\
↪net4\System.Management.Automation.dll</HintPath>
          <Private>False</Private>
        </Reference>
```

14.3.1.1 Setup `kenaflow.demoplugin.v1`

The project contains a demo workflow that uses the cmdlet `Get-PluginTestV1` provided by the plugin. - The demo workflow is of type "POWERSHELL" and is located in the subfolder `_ready_workflow` in the project.

After you have patched all project files you can open the solution in Visual Studio (2017 / 2019) *Community Edition* (https://visualstudio.microsoft.com/downloads/).

When you compile the project its output is written to the `Plugins` subfolder of *kenaflow*, e.g. `C:\Program Files\kenaflow\Plugins`.

It's easy to debug the plugin.

Just set the startup options:

`Start External Program` => path to `kenaflow.exe`

`Command line arguments => --exec --workflow "<path-to-_ready_workflow>" --notbe --noignore`

Now you can debug the custom cmdlet.

If you want to run the plugin from within a workflow debug session - e.g. with Visual Studio Code or PowerShell ISE - you can use this settings:

`Start External Program => path to C:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe`

`Command line arguments => "c:\source\kenaflow.demoplugin.v1\_ready_workflow\script.ps1"`

Here you need to use the script file path as single argument. The path is case sensitive - otherwise breakpoints in PowerShell ISE will not work!

Continue here to get more details about *Plugin Architecture Version 1*.

14.3.1.2 Setup `kenaflow.demoplugin.v2`

The project contains a custom workflow type "DEMO2" with SubType "SUB1" and a custom cmdlet `Get-PluginTestV2`.

There is a demo workflow of type "DEMO2::SUB2" and is located in the subfolder `_ready_workflow` in the project folder.

After you have patched all project files you can open the solution in Visual Studio (2017 / 2019) *Community Edition* (https://visualstudio.microsoft.com/downloads/).

When you compile the project its output is written to the `Plugins` subfolder of *kenaflow*, e.g. `C:\Program Files\kenaflow\Plugins`.

It's easy to debug the plugin.

Just set the startup options:

`Start External Program` => path to `kenaflow.exe`

`Command line arguments => --exec --workflow "<path-to-_ready_workflow>" --notbe --noignore`

Now you can debug the custom workflow type.

If you want to run the plugin from within a workflow debug session - e.g. with Visual Studio Code or PowerShell ISE - you can use this settings:

`Start External Program => path to C:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe`

`Command line arguments => "c:\source\kenaflow.demoplugin.v1\_ready_workflow\script.ps1"`

Here you need to use the script file path as single argument. The path is case sensitive - otherwise breakpoints in PowerShell ISE will not work!

Continue here to get more details about *Plugin Architecture Version 2*.

14.4 Plugin Architecture Version 1

The following C# class is taken from our demo project https://github.com/ikarstein/kenaflow.demoplugin.v1

```
    using System;
    using System.Collections;
    using System.Collections.Generic;
    using System.Management.Automation;

    namespace kenaflow.demoplugin.v1
    {
        public class DemoPluginV1 : KenaflowPluginV1
        {
            private readonly KenaflowPluginConfigurationV1 MyConfig = new␣
↪KenaflowPluginConfigurationV1
            {
                Id = Guid.Parse("{E9132626-B03B-415A-A73F-07CBF56D151E}"),
                Enabled = true,
                Name = "Demo Plugin v1",
                Version = new Version("1.0.0.0"),
                MinKenaflowVersion = new Version("4.0.0.0"),
                LogDebug = true,
                            SupportedWorkflowTypes = new List<string>{"*"}
            };
```

(continues on next page)

(continued from previous page)

```
            protected override KenaflowPluginConfigurationV1␣
↪GetPluginConfig(Options options)
            {
                //Return the Plugin Configuriation to kenaflow.
                LogNormal("Loading config of DemoPlugin v1");
                return MyConfig;
            }

            protected override void Loaded()
            {
                //Can be used to initialize the plugin after it was loaded and␣
↪successfully registered.
            }

            protected override void Init(WFConfig config)
            {
                MyConfig.Enabled = true;
            }
        }

        [Cmdlet("Get", "PluginTestV1")]
        public class Test : PSCmdlet
        {
            protected override void ProcessRecord()
            {
                WriteObject($"Test: {DateTime.Now}");
            }
        }
    }
```

The plugin class is derrived from `KenaflowPluginV1`.

! This base class offers some tools that are described in this article: *Helper Methods*.

It has a configuration object that is stored in a private property called `MyConfig`.

During initialization *kenaflow* requests this configuration object by calling `GetPluginConfig`. The parameter `options` will contain used the command line options for `kenaflow.exe`.

`Id` is the unique ID of the plugin.

`Enabled` tells *kenaflow* whether the plugin is available. If set to `false` the plugin is going to be ignored. **But if** `kenaflow.exe` was started with command line switch `--plugin <assemlby>` the plugin is executed even if it's `Enabled`-setting is ` false.

`Name` is the display name of the plugin. It must be unique in *kenaflow*. No other plugin can have the same name!

`Version` is the version of the plugin. It's the developers choice what to set here.

`MinKenaflowVersion` is the minimum *kenaflow* version that is supported by the plugin.

`LogDebug` tells *kenaflow* to write some debugging messages to the log reporting plugin method calls.

`SupportedWorkflowTypes` can be used to restrict the provided Cmdlets to certain workflow types. Valid values are:

- `SHAREPOINT`
- `SHAREPOINT::SP2013` or `SHAREPOINT::SP2013::ALL` or `SHAREPOINT::SP2013::SITE` or `SHAREPOINT::SP2013::LASTMODIFIED` or `SHAREPOINT::SP2013::STATEMACHINE` or `SHAREPOINT::SP2013::QUERY`
- `SHAREPOINT::SP2016` or `SHAREPOINT::SP2016::ALL` or `SHAREPOINT::SP2016::SITE` or `SHAREPOINT::SP2016::LASTMODIFIED` or `SHAREPOINT::SP2016::STATEMACHINE` or `SHAREPOINT::SP2016::QUERY`

- SHAREPOINT::SP2019 or SHAREPOINT::SP2019::ALL or SHAREPOINT::SP2019::SITE or SHAREPOINT::SP2019::LASTMODIFIED or SHAREPOINT::SP2019::STATEMACHINE or SHAREPOINT::SP2019::QUERY
- SHAREPOINT::SPO or SHAREPOINT::SPO::ALL or SHAREPOINT::SPO::SITE or SHAREPOINT::SPO::LASTMODIFIED or SHAREPOINT::SPO::STATEMACHINE or SHAREPOINT::SPO::QUERY
- SHAREPOINT::ALL
- SHAREPOINT::SITE
- SHAREPOINT::LASTMODIFIED
- SHAREPOINT::STATEMACHINE
- SHAREPOINT::QUERY
- EMAIL
- EMAIL::EXCHANGE2007SP1
- EMAIL::EXCHANGE2010
- EMAIL::EXCHANGE2010SP1
- EMAIL::EXCHANGE2010SP2
- EMAIL::EXCHANGE2013
- EMAIL::EXCHANGE2013SP1
- EMAIL::EXCHANGE
- EMAIL::EXCHANGEONLINE
- EMAIL::IMAP
- EMAIL::POP3
- POWERSHELL
- ... and all custom workflow types that are implemented by Plugins of architecture version 2.

To support **all** workflow types please specify:

```
  SupportedWorkflowTypes = new List<string>{"*"}
```

```
> To support **no** workflow types please specify:
> ```cs
    SupportedWorkflowTypes = null
```

or

```
  SupportedWorkflowTypes = new List<string()
```

```
#### Method `KenaflowPluginConfigurationV1 GetPluginConfig(Options options)`

The method is called to request the plugin configuration.

#### Method `void Loaded()`

The method is called immediately after the plugin was registered successfully.
```

(continues on next page)

(continued from previous page)

```
#### Method `void Init(WFConfig config)`

The method `Init` is called BEFORE each workflow run. Here the plugin gets the␣
↪configuration of the currently executed workflow. The plugin can decide to␣
↪deactivate itself by setting `MyConfig.Enabled` to `false`.


### Cmdlet Definition

The class `Test` in the exmaple above defines the cmdlet `Get-PluginTestV1`.



<!--\pages\08.Plugins\03.PluginArch2\default.md-->

## Plugin Architecture Version 2
<a name='ec6b040d-d502-4f07-910a-1fec10915158'/>

The following C# class is taken from our demo project https://github.com/ikarstein/
↪kenaflow.demoplugin.v2

```cs
  using System;
  using System.Collections;
  using System.Collections.Generic;
  using System.Management.Automation;

  namespace kenaflow.demoplugin.v2
  {
      public class DemoPluginV2 : KenaflowPluginV2
      {
          private readonly KenaflowPluginConfigurationV2 MyConfig = new␣
↪KenaflowPluginConfigurationV2
          {
                          Id = Guid.Parse("{BC3E435B-ADD6-4764-B415-3D887403222C}
↪"),
              Enabled = true,
              Name = "Demo Plugin v2",
              Version = new Version("1.0.0.0"),
              MinKenaflowVersion = new Version("4.0.0.0"),
              PSModule = true,
              CustomWorkflowType = "DEMO2::SUB1",
              LogDebug = true,
                          SupportedWorkflowTypes = new List<string>{"DEMO2::SUB1"}
          };

          private string SomePrivateData { get; set; }

          /* BEGIN Accessible from PowerShell */
          public KenaflowPluginConfigurationV2 GetPluginConfig()
          {
              return MyConfig;
          }

          public WFConfig GetCurrentWorkflowConfig()
          {
              return WorkflowConfig;
          }
```

(continues on next page)
```

(continued from previous page)

```
        /* END Accessible from PowerShell */

        protected override KenaflowPluginConfigurationV2 GetPluginConfig(Options␣
↪options)
        {
            //Return the Plugin Configuriation to kenaflow.
            LogNormal("Loading config of DemoPlugin v1");
            return MyConfig;
        }

        protected override void Loaded()
        {
            //Can be used to initialize the plugin after it was loaded and␣
↪successfully registered.
        }

        protected override string CheckConfig(WFConfig config)
        {
            //Can be used to check the loaded workflow config (_wfconfig.ps1)
            //To return an error just return a string. On success return NULL.
            return null;
        }

        protected override void Init(WFConfig config)
        {
            //Can be used to initialize the current workflow run.
        }

        protected override object Connect()
        {
            //Connect to your system. Return a handle to the connection.␣
↪Whatever you want.
            return null;
        }

        protected override string GetLastProcessedItemId(object li)
        {
            //In case of an aborted workflow this is called to get the ID of the␣
↪last processed item. The value is later passed to 'GetItems'
            return li as string;
        }

        protected override void PreProcessEventData(ref Hashtable␣
↪hashtableEventData)
        {
            //In case of remote events this is method can be used to modify the␣
↪remote data BEFORE initiating the workflow.
        }

        protected override void PreProcessWorkflowConfigValues1(ref Dictionary
↪<string, object> currentWorkflowConfigValues)
        {
            //after config values are read from _wfconfig.ps1
        }

        protected override void PreProcessWorkflowConfigValues2(ref Dictionary
↪<string, object> currentWorkflowConfigValues)
        {
            //after config values are modified by engine BEFORE execution
        }
```

(continues on next page)

(continued from previous page)

```
        protected override bool ProcessLastRun(DateTime lastRun, ref string␣
↪errorMessage, ref string errorCode)
        {
            return true; // TRUE => do not execute workflow.
        }

        protected override bool ProcessTBEandCRON(ref DateTime dt)
        {
            return true; // TRUE => do not execute workflow. Next runtime can be␣
↪set in 'dt';
        }

        protected override List<object> GetItems(Hashtable eventData, string␣
↪lastProcessedItem)
        {
            //Get list of items for processing by workflow script.
            return new List<object> { 1, 2, 3, 4, 5 };
        }

        protected override void PreItemProcessing(ref object li)
        {
            SerializeStorageData(Guid.Parse(WorkflowConfig.Id), $"key_{li}", new
↪{ a = "a", b = "b" }, false, "");
        }

        protected override void PostItemProcessing(ref object li)
        {
            var d = DeserializeStorageData(Guid.Parse(WorkflowConfig.Id), $"key_
↪{li}", "");
            RemoveStorageData(Guid.Parse(WorkflowConfig.Id), $"key_{li}");
        }

        protected override void CleanupAfterItem()
        {
            //Called after the workflow has processed all items. (Also in case␣
↪of failure.)
        }

        protected override void CleanupAfterRun()
        {
            //Called after the workflow has processed all items.
        }

        protected override void ItemRemoval(object li, int removalState)
        {
            //Called at the end of the item processing.
            // 0 = no removal requested
            // 1 = removal requested
            // 2 = move to recycle bin requested
            base.ItemRemoval(li, removalState);
        }

        protected override void Disconnect(object connection)
        {
            //Disconnect from your system. You get to handle that was returned␣
↪by 'Connect'.
            //Only called at the end of the workflow run if "Connect" returns a␣
↪value != null.
        }

        /**** TOOLS ****/
```

(continues on next page)

(continued from previous page)

```
        protected override object GetData(string key, object item)
        {
            //can be used to read persisted data. 'item' can be null, e.g. if␣
↪using Set-KFData
            return null;
        }

        protected override void SetData(string key, object item, object data)
        {
            //can be used to persist data. 'item' can be null, e.g. if using Set-
↪KFData
        }

        protected override void RemoveData(string key, object item)
        {
            //can be used to remove persisted data. 'item' can be null, e.g. if␣
↪using Set-KFData
        }

        protected override bool HandleException(Exception exception)
        {
            return true; // => throw exception in engine. FALSE => don't throw
        }

        protected override void RegisterReceiver(string url)
        {
            //Can be used to register web hooks.
        }

        protected override void UnregisterReceiver(string url)
        {
            //Can be used to un-register web hooks.
        }
    }

    [Cmdlet("Get", "PluginTestV2")]
    public class Test : PSCmdlet
    {
        protected override void ProcessRecord()
        {
            WriteObject($"Test: {DateTime.Now}");
        }
    }
}
```

The plugin class is derrived from `KenaflowPluginV2`.

! This base class offers some tools that are described in this article: *Helper Methods*.

It has a configuration object that is stored in a private property called `MyConfig`.

During initialization *kenaflow* requests this configuration object by calling `GetPluginConfig`. The parameter `options` will contain used the command line options for `kenaflow.exe`.

`Id` is the unique ID of the plugin.

`Enabled` tells *kenaflow* whether the plugin is available. If set to `false` the plugin is going to be ignored. **But if** `kenaflow.exe` was started with command line switch `--plugin <assemlby>` the plugin is executed even if it's `Enabled`-setting is ` false.

`Name` is the display name of the plugin. It must be unique in *kenaflow*. No other plugin can have the same name!

`Version` is the version of the plugin. It's the developers choice what to set here.

`PSModule` tells *kenaflow* whether the plugin contains custom cmdlets or not.

`CustomWorkflowType` tells *kenaflow* that the plugin contains a custom workflow type and what the name of the workflow type is. The custom workflow type can have a sub type. Type and sub type are seperated by `::`.

`MinKenaflowVersion` is the minimum *kenaflow* version that is supported by the plugin.

`LogDebug` tells *kenaflow* to write some debugging messages to the log reporting plugin method calls.

`SupportedWorkflowTypes` can be used to restrict the provided Cmdlets to certain workflow types. Valid values are:

- `SHAREPOINT`
- `SHAREPOINT::SP2013` or `SHAREPOINT::SP2013::ALL` or `SHAREPOINT::SP2013::SITE` or `SHAREPOINT::SP2013::LASTMODIFIED` or `SHAREPOINT::SP2013::STATEMACHINE` or `SHAREPOINT::SP2013::QUERY`
- `SHAREPOINT::SP2016` or `SHAREPOINT::SP2016::ALL` or `SHAREPOINT::SP2016::SITE` or `SHAREPOINT::SP2016::LASTMODIFIED` or `SHAREPOINT::SP2016::STATEMACHINE` or `SHAREPOINT::SP2016::QUERY`
- `SHAREPOINT::SP2019` or `SHAREPOINT::SP2019::ALL` or `SHAREPOINT::SP2019::SITE` or `SHAREPOINT::SP2019::LASTMODIFIED` or `SHAREPOINT::SP2019::STATEMACHINE` or `SHAREPOINT::SP2019::QUERY`
- `SHAREPOINT::SPO` or `SHAREPOINT::SPO::ALL` or `SHAREPOINT::SPO::SITE` or `SHAREPOINT::SPO::LASTMODIFIED` or `SHAREPOINT::SPO::STATEMACHINE` or `SHAREPOINT::SPO::QUERY`
- `SHAREPOINT::ALL`
- `SHAREPOINT::SITE`
- `SHAREPOINT::LASTMODIFIED`
- `SHAREPOINT::STATEMACHINE`
- `SHAREPOINT::QUERY`
- `EMAIL`
- `EMAIL::EXCHANGE2007SP1`
- `EMAIL::EXCHANGE2010`
- `EMAIL::EXCHANGE2010SP1`
- `EMAIL::EXCHANGE2010SP2`
- `EMAIL::EXCHANGE2013`
- `EMAIL::EXCHANGE2013SP1`
- `EMAIL::EXCHANGE`
- `EMAIL::EXCHANGEONLINE`
- `EMAIL::IMAP`
- `EMAIL::POP3`
- `POWERSHELL`
- ... and all custom workflow types that are implemented by Plugins of architecture version 2.

  To support **all** workflow types please specify:

```
SupportedWorkflowTypes = new List<string>{"*"}
```

```
> To support **no** workflow types please specify:
> ```cs
    SupportedWorkflowTypes = null
```

or

```
  SupportedWorkflowTypes = new List<string()
```

```


#### Method `KenaflowPluginConfigurationV2 GetPluginConfig(Options options)`

The method is called to request the plugin configuration.

#### Method `void Loaded()`

The method is called immediately after the plugin was registered successfully.

### Tool Methods

The following methods are used by _kenaflow_ to initialize the plugin or request␣
↪special actions.

#### Method `string CheckConfig(WFConfig config)`

This method can be used to check the loaded workflow config (`_wfconfig.ps1`).

To return an error just return a `string`.

On success return `NULL`.

#### Method `string GetLastProcessedItemId(object li)`

A workflow has a maximum execution time. BEFORE each item processing the elapsed␣
↪time of the current workflow run is checked. If the time exceeded the limit the␣
↪workflow execution is stopped. Therefore a hint to the next item is stored in␣
↪the `_wfdata` folder.

The get the hint this method is called. The hint is returned as string. On the␣
↪next workflow run its passed to the method `GetItems`.

#### Method `object GetData(string key, object item)`

This method is called if the workflow uses [`Get-KFData`](#f6012b68-c25c-4ef8-b821-
↪7819494fdded) or [`Get-KFItemData`](#5878aa25-2b8d-4973-bee3-e4f9b8abbb9c).

The plugin developer decides that to do with the data.

`key` and `item` are both used as key to the data. `item` represents the currently␣
↪processed item - if any.

#### Method `void SetData(string key, object item, object data)`
This method is called if the workflow uses [`Set-KFData`](#c9b420f8-0645-4df0-9714-
↪88a291bb3704) or [`Set-KFItemData`](#6a1ef7fa-6ad2-4f9c-8310-eea754d1cefe).

The plugin developer decides that to do with the data.

`key` and `item` are both used as key to the data. `item` represents the currently␣
↪processed item - if any.
```

(continues on next page)

(continued from previous page)

```
The object in `data` should be serialized.

#### Method `void RemoveData(string key, object item)`

This method is called if the workflow uses [`Remove-KFData`](#29c2a41e-57d4-4811-
↪a416-df2ffc648555) or [`Remove-KFItemData`](#1a0e43ac-17d3-4a96-a2d0-
↪eca71781c022).

`key` and `item` are both used as key to the data. `item` represents the currently␣
↪processed item - if any.

`void RegisterReceiver(string url)`
      {
          //Can be used to register web hooks.
      }

`void UnregisterReceiver(string url)`
      {
          //Can be used to un-register web hooks.
      }


### Flow Methods

The following methods are used by _kenaflow_ to interact with the plugin and␣
↪enable the plugin to execute.

The methods are executed in a fixed order during each workflow run. Not all are␣
↪executed every time.

This is the order of the methods:

1. `Init`
1. `PreProcessEventData`
1. `CheckLastRun`
1. `CheckTBEandCRON`
1. `PreProcessWorkflowConfigValues1`
1. `Connect`
1. `RegisterReceiver` or `UnregisterReceiver`
1. `GetItems`
> If the result is not `null` not an empty list of objects (`System.Collections.
↪Generic.List<object>`) the next step is executed.
>
> Otherwise the **workflow script** is executed next and afterwards␣
↪`CleanupAfterRun`.
1. `HandleException` ... __*if*__ an **unhandled exception** occurs between the␣
↪next step and `PostItemProcessing` the method `CleanupAfterItem`.
1. `PreItemProcessing`
1. Now the workflow script is executed!
1. `PostItemProcessing`
1. `ItemRemoval`
1. `CleanupAfterItem` ... loop back to `PreItemProcessing`
1. `CleanupAfterRun`
1. `Disconnect`


#### Method `void Init(WFConfig config)`

`Init` is called BEFORE the execution of a workflow. It's called for every loaded␣
↪and enabled workflow! This could be used to manipulate the workflow␣
↪configuration `config`.
```

(continues on next page)

(continued from previous page)

```
AFTER the workflow processing was started but BEFORE executing the workflow script␣
↪the `Connect` method is called. This can be used to connect to a third party␣
↪system. The method schould return `null` if no connection is needed OR␣
↪`Disconnect` is not required.

#### Method `void PreProcessEventData(ref Hashtable hashtableEventData)`

In case of remote events this is method can be used to modify the remote data␣
↪BEFORE initiating the workflow.

#### Method `bool CheckLastRun(ref DateTime lastRun, ref string errorMessage, ref␣
↪string errorCode)`

Before _kenaflow_ executes a workflow it checks the last run of the workflow.

First _kenaflow_ loads the last execution state file from disk (from folder `_
↪wfdata`). There can be an error reading and interpreting the file. If an error␣
↪occurs the method `CheckLastRun` is called. You get the read last run timestamp␣
↪(if any) and the error message/code. All this information can be modified.

If the method returns `false` the workflow is **NOT** executed!

#### Method `bool CheckTBEandCRON(ref DateTime dt, bool checkResult)`

After successfully loading the last run timestamp it is checked against workflow␣
↪settings `TBE` ("time between execution") and `CRON`.

The check result can force the workflow run to stop.

With method `CheckTBEandCRON` the check result can be modified.

`dt` is the last run timestamp.

`checkResult` is the result of the `TBE` and `CRON` check.
+ if it is `false` the workflow should not run based on `TBE` and `CRON`. The␣
↪proposed next run time is given in `dt`.
+ if it is `true` the workflow should run.

You can modify the next run time in `dt` and you can return `true` (run workflow)␣
↪or `false` (stop workflow).

#### Method `void PreProcessWorkflowConfigValues(ref Dictionary<string, object>␣
↪currentWorkflowConfigValues)`

This method is called after loading the workflow configuration values from `_
↪wfconfig.ps1`.

The plugin can modify the workflow configuration values.


#### Method `List<object> GetItems(Hashtable eventData, string lastProcessedItem)`

This methods is called the determine the items for processing in the current␣
↪workflow run.

The items are returned as list of object (`System.Collections.Generic.List<object>
↪`).

The parameter `eventData` contains data if the workflow was started by receiving a␣
↪remote event. The data can be modified.
```

(continues on next page)

(continued from previous page)

```
The parameter `lastProcessedItem` contains the hint from the last workflow run if␣
↪this was stopped before processing all given items. Otherwise this string is␣
↪null.

If the method returns `null` or an empty list of objects the **next step is the␣
↪execution of the workflow script**.

#### Method `bool HandleException(Exception exception)`

If an unhandled exception occurs during execution of the workflow script this␣
↪method is called.

If the workflow has an error handling script ([Error Handling](#569c5e49-2032-4248-
↪8083-0fbecffb4e4c)) this is called first. If this script handles the exception␣
↪successfully (return "OKAY") the method `HandleException` is not executed.

If it returns `false` the exception is **NOT** throws by the engine.

if it returns `true` the exception is thrown by the engine.


#### Method `void PreItemProcessing(ref object li)`

This method is called before execting workflow script for the current item.

#### Method `void PostItemProcessing(ref object li)`

This method is called after the execution of the workflow script for the current␣
↪item.

#### Method `void ItemRemoval(object li, int removalState)`

After the item execution this method can be used to remove the item if required.

The workflow developer handles the removal. The method just indicates the␣
↪requested removal state:

+ `0` = no removal requested
+ `1` = removal requested
+ `2` = move to recycle bin requested

#### Method `void CleanupAfterItem()`

This method is called after the processing of the current item is finished. It is␣
↪executed also in case of failure.
      }

#### Method `void CleanupAfterRun()`

The method is executed after the workflow has processed all items (if `GetItems`␣
↪did not return `null` or an empty list of objects) or after the workflow script␣
↪was executed (if `GetItems` returned `null` or an empty list of objects).

#### Method `void Disconnect(object connection)`

This can be used to disconnect from the third-party system.

It is only called if `Connect` returned an object ( = not `null`). This object is␣
↪passed to `Disconnect`.
```

(continues on next page)

(continued from previous page)

```
### Cmdlet Definition

The class `Test` in the exmaple above defines the cmdlet `Get-PluginTestV2`.




<!--\pages\08.Plugins\04.PluginTools\default.md-->

## Helper Methods
<a name='e76f3878-ba94-4090-b925-0a64b87f3834'/>

The base classes `KenaflowPluginV1` and `KenaflowPluginV2` for custom plugin␣
↪development offer some tools in form of methods and properties. They can be used␣
↪within the plugin.

### Properties

`WFConfig WorkflowConfig { get; }`
> As soon as the current workflow is determined it's configuration is available in␣
↪the plugin.

`KenaflowPluginConfigurationV1 PluginConfig { get; }`
`KenaflowPluginConfigurationV2 PluginConfig { get; }`
> The property returns the plugin configuration.

`dynamic CustomData { get; }`
> This is a shortcut to `WorkflowConfig.CustomDataDynamic`.


### Configuration Access Methods

`T GetConfig<T>(string key, T Default)`

> Can be used to read settings from the global configuration of kenaflow.
>
> By default it is not allowed to read the following settings:
> + ENCRYPTIONPASSPHRASE
> + LICENSINGMAILADDRESS
> + SERIAL
> + ACTIVATIONCODE
> + LASTACTIVATIONCODE
> + LASTOFFINEDEACTIVATIONCODE


### Storage Methods

`SetDBData(string key, object data)`
`SetDBData(Guid workflowId, string key, object data)`
> Can be used to *serialize* objects to the SQLite database of  _kenaflow_. - If␣
↪the first syntax is used the Id of the currently processed workflow is used! -␣
↪Using the second syntax with a workflow ID it's possible to access data written␣
↪by another workflow!

`object GetDBData( string key)`
`object GetDBData(Guid workflowId, string key)`
> Can be used to read / deserialize objects from the SQLite database of _kenaflow_.
↪ - If the first syntax is used the Id of the currently processed workflow is␣
↪used! - Using the second syntax with a workflow ID it's possible to access data␣
↪written by another workflow!
```

(continues on next page)

(continued from previous page)

```
`void RemoveDBData( string key)`
`void RemoveDBData(Guid workflowId, string key)`
> Can be used to remove objects from the SQLite database of  _kenaflow_. - If the␣
↪first syntax is used the Id of the currently processed workflow is used! - Using␣
↪the second syntax with a workflow ID it's possible to access data written by␣
↪another workflow!

`void SetStorageData(bool sessionStorage, string key, byte[] data, bool encrypt,␣
↪string additionalPassphrase="")`
`void SetStorageData(bool sessionStorage, Guid workflowId, string key, byte[] data,␣
↪bool encrypt, string additionalPassphrase="")`
> Can be used to write binary data to the permanenet storage or session storage of␣
↪ _kenaflow_. The data is written to disk. It can be encrypted. An additional␣
↪passphrase can be provided. ("Additional" means that it's used in conjunction␣
↪with the configured passphrase from the global configuration. The developer does␣
↪not know the complete passphrase, only his/her "additional" part.)
> If the first syntax is used the Id of the currently processed workflow is used! -
↪ Using the second syntax with a workflow ID it's possible to access data written␣
↪by another workflow!

`(bool found, byte[] data) GetStorageData(bool sessionStorage, string key, string␣
↪additionalPassphrase="")`
`(bool found, byte[] data) GetStorageData(bool sessionStorage, Guid workflowId,␣
↪string key, string additionalPassphrase="")`
> Used to read binary data from the permanent storage or session storage of _
↪kenaflow_. If the data was encrypted with `SetStorageData` with an additional␣
↪passphrase it must be specified here.
> If the first syntax is used the Id of the currently processed workflow is used! -
↪ Using the second syntax with a workflow ID it's possible to access data written␣
↪by another workflow!

`void SerializeStorageData(bool sessionStorage, string key, object data, bool␣
↪encrypt, string additionalPassphrase="")`
`void SerializeStorageData(bool sessionStorage, Guid workflowId, string key, object␣
↪data, bool encrypt, string additionalPassphrase="")`
> Can be used to serialize data to the permanenet storage or session storage of _
↪kenaflow_. The data is written to disk. It can be encrypted. An additional␣
↪passphrase can be provided. ("Additional" means that it's used in conjunction␣
↪with the configured passphrase from the global configuration. The developer does␣
↪not know the complete passphrase, only his/her "additional" part.)
> If the first syntax is used the Id of the currently processed workflow is used! -
↪ Using the second syntax with a workflow ID it's possible to access data written␣
↪by another workflow!

`object DeserializeStorageData(bool sessionStorage, string key, string␣
↪additionalPassphrase="")`
`object DeserializeStorageData(bool sessionStorage, Guid workflowId, string key,␣
↪string additionalPassphrase="")`
> Used to deserialize objects from the permanent storage or session storage of _
↪kenaflow_. If the data was encrypted with `SetStorageData` with an additional␣
↪passphrase it must be specified here.
> If the first syntax is used the Id of the currently processed workflow is used! -
↪ Using the second syntax with a workflow ID it's possible to access data written␣
↪by another workflow!

`void RemoveStorageData(bool sessionStorage, string key)`
`void RemoveStorageData(bool sessionStorage, Guid workflowId, string key)`
> Can be used to remove binary data or serialized objects from the permanenet␣
↪storage or session storage of _kenaflow_.
> If the first syntax is used the Id of the currently processed workflow is used! -
↪ Using the second syntax with a workflow ID it's possible to access data written␣
↪by another workflow!
```

(continues on next page)

(continued from previous page)

```
### Logging
`Log(LogLevel logLevel, string message)`
> Logging.

`void LogNormal(string message)`
> Shortcut to log a "normal" message. This is the first "okay" log level.

`void LogDebug(string message)`
> Shortcut to log a "debug" message. This is the lowest log level! By default␣
↪messages of this type are not recorded. The log level for log files can be␣
↪changed in the global configuration.

`void LogVerbose(string message)`
> Shortcut to log a "verbose" message. This is the second log level.

`void LogError(string message)`
> Shortcut to log an "error" message. This errors are "not critical".

`void LogCritical(string message)`
> Shortcut to lot an "critical" error message. Errors of this type indicate an␣
↪application failure. Administrators should react immediately.

`void LogException(Exception exception)`
> Used to log exception objects.


<!--\pages\09.ScriptReference\default.md-->

# PowerShell Reference
<a name='9333ce08-75b2-49c5-80de-c6f13a17a549'/>


_kenaflow_ provides a set of PowerShell Cmdlets that allow you to perform␣
↪frequently recurring tasks in your workflow scripts.

Here is an introduction to the syntax of PowerShell:

[Introduction and Syntax](#20ae52e4-630b-42f5-a9e2-7df3c5f8ca82)

Here is an overview of the _kenaflow_ cmdlets:

[kenaflow Cmdlets](#3e54999d-7d3a-4f75-aa4e-0828575c9f86)


<!--\pages\09.ScriptReference\01.Syntax\default.md-->

## Introduction and Syntax
<a name='20ae52e4-630b-42f5-a9e2-7df3c5f8ca82'/>

_kenaflow_ provides a set of PowerShell Cmdlets that allow you to perform␣
↪frequently recurring tasks in your workflow scripts.

The Cmdlets are designed for the _kenaflow_ runtime environment and can not be␣
↪used without them, with a few exceptions.

In the following descriptions we use the following syntax:

```powershell
  AnyKindOf-PowerShellCommand
          [-optionalParameterName] <parameter-value>
```

(continues on next page)
```

(continued from previous page)

```
        [-optionalParameter <parameter-value> (<default-value>)]
        [-optionalSwitch[:<default-value>]]
        [-parameterWithConstantValues Value1 <or> Value2 <or> Value3]
```

The PowerShell-Cmdlet is called `AnyKindOf-PowerShellCommand`.

This is followed by a mandatory parameter whose name `-optionalParameterName` can be omitted. In this case, only the `<parameter-value>` has to be specified.

If there are several parameters with an optional name, then the order of their `<parameter values>` must be strictly adhered to!

The following is a fully optional parameter that has the name `-optionalParameter`. If this parameter is not specified, the specified (`<default-value>`) will be used. The default value may be a concrete value or a description, such as (the current list item).

If the parameter name of an optional parameter is specified, a `<parameter-value>` must also be specified.

In PowerShell there are "switches" that have no actual parameter-value but only exist or do not exist. In PowerShell, for example, there are the much-used switches `-Verbose` or `-Confirm`.

A switch always has a default value that is either `$true` or `$false`. If the switch parameter is omitted, this default value is used.

If the switch is specified, its value is automatically `$true`.

To change the default value of a switch from `$true` to `$false`, the following lines need to be written in PowerShell:

```
    Remove-Item "...any file..." -Confirm:$false
```

The Remove-Item command always reassures if a file should really be deleted before deleting it. The `-Confirm` switch is set to `$true` by default. In order to execute the deletion of the file without enquiry, `-Confirm` must be set to `$false`.

Some PowerShell parameters have fixed presets or options that can be specified as `<parameter-value>`. The possible values are specified after the parameter, for example:

```
    [-parameterWithConstantValues Value1 <or> Value2 <or> Value3]
```

In a workflow script, "Value1" or "Value2" or "Value3" can be used.

For some Cmdlets there are several so-called "parameter sets". This means that some parameters may only be used in combination with certain others. If a parameter from a certain parameter set is used, the parameters from other parameter sets are not permitted. An error is thrown and the Cmdlet is not executed.

There may be parameters that apply to multiple parameter sets or independent of parameter sets.

The syntax for such Cmdlets is usually to write these sets as multiple lines. All parameters are specified, including those that are valid for all parameter sets or for several.

14.4.1 Error handling

All *kenaflow* specific Cmdlets have the optional parameter `[-ThrowErrors[:$true]]`

This parameter can be used to catch any errors that may occur during the execution of each Cmdlet, so that the execution of the PowerShell script is not hindered.

If you do not specify the switch with the value `$false`, the errors are passed on to the PowerShell runtime, which can cause the workflow script to break down without any other error handling.

e.g.:

```
Get-KFUserInfo -UserName 'kenaflow' -ThrowErrors:$false
```

If an error occurs in this example, then it can be queried with `Get-KFLastResult`. The return of the `Get-KFLastResult` Cmdlet then contains an Exception object.

At the level of each workflow script, the `ThrowErrors` behavior can be set:

```
Set-KFThrowErrorsPreference [-ThrowErrorPreference] $true <or> $false
```

Although by default -ThrowErrors is set to `$true`, this standard can be overridden with Set-KFThrowErrorPreference. When used, this new default setting applies until the end of the current script or until the next use of `Set-KFThrowErrorsPreference`.

e.g.:

```
Set-KFThrowErrorsPreference $false
Get-KFUserInfo -UserName 'kenaflow'
$e = Get-KFLastResult
```

Here, the default setting for `-ThrowErrors` of Get-KFUserInfo is set to `$false` by `Set-KFThrowErrorPreference`. So no error is thrown if the user does not exist.

The value can be overridden for each Cmdlet. e.g.:

```
Set-KFThrowErrorsPreference $false

#some Code

Get-KFUserInfo -UserName 'kenaflow' -ThrowErrors:$true
```

An error would be thrown here if the user 'kenaflow' does not exist.

At workflow configuration level in `_wfconfig.ps1`, the `ThrowErrorsPreference = $true` parameter can be used to set the default value for the entire workflow and all workflow scripts in the workflow. This value can later be overridden with `Set-KFThrowErrorsPreference` in the script or with the `-ThrowErrors` parameter for a Cmdlet.

14.5 kenaflow Cmdlets

14.5.1 Basic Cmdlets

- `Get-KFLastResult`

  Returns the data that the last *kenaflow* cmdlet returned. If the last *kenaflow* cmdlet failed and `-ThrowErrors:$false` was set, this cmdlet returns the exception object.

- `Start-KFSharePointModule`

  This can be used to start the SharePoint PnP PowerShell module with the correct SharePoint platform assemblies. This open source project delivers lots of helpful SharePoint related cmdlets. The module is embedded in *kenaflow*.

- *`Write-KFLog`*

  Can be used to write messages to the *kenaflow* log.

- *`Write-KFLogOutput`*

  Can be used to serialize objects to the *kenaflow* log.

Available only for SharePoint workflows:

- *`Write-KFHistory`*

14.5.2 Debugging and Runtime

- *Get-KFInDebug*

  Used to get a hint whether the script is currently debugged.

- *Invoke-KFException*

  Can be used to throw an exception from within a *kenaflow* cmdlet. The *kenaflow* cmdlet common parameter `-ThrowErrors` is used and therefore the current setting of *Set-KFThrowErrorsPreference*.

- *Invoke-Kenaflow*

  Used to initialize *kenaflow* for debugging.

- *Invoke-KFDebugBreak*

  Hard coded breakpoint.

- *Invoke-KFUnstructuredDataEventSimulation*

  Used to simulate an "unstructured data" remote event.

- *Invoke-KFLinkEventSimulation*

  Used to simulate an "link" remote event.

- *Invoke-KFCustomDataEventSimulation*

  Used to simulate an "custom data" remote event.

- *Invoke-KFSharePointEventSimulation*

  Used to simulate an "SharePoint" remote event.

- *Invoke-KFCtrlC*

  Simulates pressing `Ctrl+C` to stop the current *kenaflow* process.

- *Invoke-KFTerminatePosh*

  Stops the current PowerShell pipeline. (For debugging and testing.)

- *Start-KFSharePointModule*

  Starts the SharePoint PnP PowerShell module for the specified SharePoint platform.

- *Get-KFLastResult*

  Returns the last result from the last executed *kenaflow* cmdlet. Also used to get the last occured exceptiong if throwing was disabled with `-ThrowErrors:$false` or *Set-KFThrowErrorsPreference*.

14.5.3 Workflow Control

- *Get-KFScriptRunTimeout*

  Get the amount of time before the current script will be terminated.

- *Set-KFScriptTimeout*

  Set the amount of time before the current script will be terminated.

- *Stop-KFScript*

  Stops the current workflow script, state or workflow.

- Get-KFThrowErrorsPreference

  Get the current error throwing preference of *kenflow*.

- Set-KFThrowErrorsPreference

Set the error throwing preference of *kenflow*.

14.5.4 E-Mails, Templates, PDF

Sending e-mails.

- *Invoke-KFTemplate*

  Replaces placeholders in a given template string.

- *Send-KFMail*

  Sends an email.

- *New-KFPDF*

  Creates a PDF from HTML.

- *Invoke-KFFileTemplate*

  Replaces placeholders in Word and PowerPoint files.

14.5.5 Excel File Tools

- *Import-KFExcelFile*

  Import Excel file

- *New-KFExcelFile*

  Create Excel file from data

- *Update-KFExcelFile*

  Update Excel file

14.5.6 Tools

- *Get-KFDateTime*

  Can be used to get the current date and time or a date and time relative to a "reference DateTime" set by `Set-KFDateTime`.

- *Set-KFDateTime*

  Set a reference DateTime. For debugging and testing.

- *Get-KFSPDatetimeLocal*

  Convert a UTC datetime value in a SharePoint list item field to the localized DateTime value as it was entered or is viewed by the user.

- *Set-KFSPDatetimeLocal*

  Convert a localized DateTime value to UTC. The command assumes that the `-DateTimeLocal` passed is in the time zone set on the SharePoint web regional settings.

- *Clear-KFDataTime*

  Clears the previous set reference DateTime. For debugging and testing.

- *Get-KFHash*

  Returns a Base64 encoded SHA512 hash.

- *ConvertFrom-KFISODateTime*

Converts a DateTime from the SharePoint used string format.

- *ConvertTo-KFISODateTime*

  Converts a given DateTime to the SharePoint used string format.

- *ConvertFrom-KFJSON*

  Converts a given JSON string to a PowerShell Hashtable.

- *ConvertoTo-KFCAML*

  Converts a given *kenaflow* query string to CAML.

- *Get-KFNormalizedCAML*

  Fixes a CAML string.

- *Get-KFCron*

  Used to get DateTimes based on a CRON pattern.

- *Get-KFPrettyXML*

  Get pretty and readable XML from a XML string.

- *New-KFBarcode*

  Creates barcode images.

- *Get-KFBarcode*

  Reads barcodes form images.

- *Protect-KFData*

  Encrypts data.

- *Unprotect-KFData*

  Decrypts data.

- *Get-KFTest*

  Gets a simple string. For testing only.

14.5.7 Data Handling

14.5.7.1 Available for all workflow types:

- *Set-KFDBData*

  Serializes data to the *kenaflow* embedded SQLite database.

- *Get-KFDBData*

  Deserializes data from the *kenaflow* embedded SQLite database.

- *Remove-KFDBData*

  Removes data to the *kenaflow* embedded SQLite database.

14.5.7.2 Available only for SharePoint workflow types:

- *Get-KFData*

  Serializes data to the workflow data list.

- *Set-KFData*

  Deserializes data from the workflow data list.

- *Remove-KFData*

  Removes data from the workflow data list.

14.5.7.3 Available only for SharePoint *list* and *state machine* workflow types:

- *Set-KFItemData*

  Serializes data in the context of the current list item to the workflow data list.

- *Get-KFItemData*

  Deserializes data in the context of the current list item from the workflow data list.

- *Remove-KFItemData*

  Removes data in the context of the current list item from the workflow data list.

14.5.8 Locking

- *Lock-KFProcess*

  Get a lock for a given key.

- *Test-KFProcessLock*

  Test a lock by a given key.

- *Unlock-KFProcess*

  Releases a lock by a given key.

14.5.8.1 SharePoint Specific Item Locking

- *Lock-KFItem*

  Locks a SharePoint list item.

- *Test-KFItemLock*

  Tests a lock on a SharePoint list item.

- *Unlock-KFItem*

  Releases a lock on a SharePoint list item.

14.5.9 SharePoint Permissions, SharePoint Principals

- *Get-KFPermission*

  Gets the permissions of a SharePoint list item.

- *Add-KFPermission*

  Adds permissions to a SharePoint list item.

- *Set-KFPermission*

  Sets the permissions on a SharePoint list item.

- *Reset-KFPermissionInheritance*

  Resets permission inheritance of a SharePoint list item.

- *Remove-KFPermission*

  Removes permissions from a SharePoint list item.

- *Stop-KFPermissionInheritance*

  Stopy permission inheritance of a SharePoint list item.

- *Get-KFUserInfo*

  Gets all profile informations from the SharePoint User Profile service application for a given user.

- *Get-KFUsers*

  Simple resolve of a given string to SharePoint user objects.

- *Resolve-KFPrincipal*

  Complex resolve of all sorts of principals, SharePoint and Active Directory.

14.5.10 SharePoint Specials

- *Set-KFContext*

  Sets the context of the current workflow execution: ClientContext, List, Item.

- Remove-KFItem

  Removes the currently processes list item after processing.

- Set-KFApproval

  Sets the SharePoint approval state.

- Set-KFFinalModerationState

  Sets the moderation state of the list item after the item was processed.

- *Get-KFLink*

  Creates a link to be used in mails to execute workflow actions if the link was clicked.

- *Get-KFUrl*

  Create some often used URLs, e.g. for the default list view or the edit form of an item.

14.5.11 Add-KFPermission

14.5.11.1 Syntax

```
Add-KFPermission
  [-Permissions] <list-of-permissions>
  [-Item <list-item>]
  [-CopyCurrentPermissions[:$false]]
  [-BreakPermissionInheritance[:$true]]
  [-ThrowResolveErrors[:$false]]
  [-ThrowErrors[:$true]]
```

14.5.11.2 Support

>= *kenaflow* 1.0.0

14.5.11.3 Return

Nothing.

14.5.11.4 Description

Use this cmdlet to add permissions to a list item.

Add means that it does not remove or replace permissions! If you plan to replace **all current** permissions with a new set of permissions please use cmdlet *Set-KFPermission*.

The `-Permissons` parameter gets a single permission object or a list of permission objects of the following form.

The permission object is specified as a PowerShell Hashtable:

```
@{
   User = <string-or-object>;
   PermissionsSets = @(<permission-set-name>[, <permission-set-name>[, ...]]);
   Quick = "CRUDMA";
)
```

The `User` property of the permission object can be:

- List of strings or objects
- string, that describe the user
- object as (array of) SharePoint `FieldUserValue`

Such a "string" of the `User` property of the permission object can be:

- a user login name
- an email address
- a SharePoint field of the currently processed or with `-Item` specified list item
- a combination of all of them separated by "," or ";"

Examples:

- `User = @($item["Author"])`
- `User = $item["Author"]`
- `User = @($item["Author"], $item["Editor"])`

- `User = "{{Author}}"`
- `User = "{{Author}},{{Editor}}"`
- `User = @($item["Author"], "{{Editor}},sharepoint\ingo", "ingo@sharepoint.farm")`

The *kenaflow* template engine is used to resolve the given strings. *Templates*

Also SharePoint groups are resolved. E.g. `User = @("Intranet Owners")`

The parameter `-PermissionSets` is a PowerShell list of one or more names of valid / existing SharePoint permission sets, such as "Full Control" or "Design". - Custom permission sets can be named too. But the have to exist!

The parameter `-Quick` is a kind of shortcut to `-PermissionSets`. *kenaflow can* create it's own permission sets as specified in the `_wfconfig.ps1` file. They have to be created using program switch *`--createpermissionsets`*. This permission sets are by default named as:

- kenaflow Create => `C`
- kenaflow Read => `R`
- kenaflow Update => `U`
- kenaflow Delete => `D`
- kenaflow Manage => `M`
- kenaflow Approve => `A`

The first letter(s) **in combination** can be used in parameter `-Quick` to specify these permission sets. - But they have to be configured in `_wfconfig.ps1` and must exist in SharePoint! - For example: `Quick="CUD"` => "Create", "Update", "Delete" - but not "Read".

The base permissions of each of the *kenaflow* defined permission sets can be found here: *Permissions*.

This is an example for a valid permission object - if the *kenaflow* permission sets are configured.

```
@{
    User = @("{{Author}}", $item["Editor"]);
    PermissionSets = @("Read");
    Quick = "CUD"
}
```

Multiple permission objects can be specified as a list:

```
@(
    @{User="sharepoint\ingo"; Quick="CUD"},
    @{User="sharepoint\kenaflow";PermissionSets=@("Full Control")}
)
```

14.5.11.5 Parameters

`[-Permissions] <list-of-permissions>`

One or more permission objects as described above.

`[-Item <list-item>]`

The list item to be processed.

`[-BreakPermissionInheritance[:$true]]`

Permissions can only be added to the list item if it has its own permissions - and not inherited permissions from its parent. If an item with active permission inheritance is given the inheritance will be stopped by default. But if you specify `-BreakPermissionInheritance:$false` the cmdlet will fail and stop if the list item has active permission inheritance.

`[-CopyCurrentPermissions[:$false]]`

If the cmdlet stop permission inheritance it can copy the permissions of the parent or start with a blank set of permissions which is the default.

`[-ThrowResolveErrors[:$false]]`

kenaflow tries to resolve the given `User` property of each permission object. If this fails and `-ThrowResolveErrors` is specified the cmdlet will fail. Otherwise the not existing or not successfully resolved principal is skipped.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.11.6 Examples

```
Add-KFPermission -Permissions @(
    @{
        User = "user1@sharepoint.farm";
        PermissionSets = @("Full Control", "Contribute");
        Quick="CRUDMA";
    },
    @{
        User = "user2@sharepoint.farm";
        PermissionSets = "Read";
        Quick="CRUDMA";
    }
)
```

```
$newEditor = "sharepoint\user3"

Add-KFPermission -Permissions @(
    @{ User="{{Author}}"; Quick="RM" },
    @{ User="{{currentEditor}}"; Quick="RM" },
    @{ User=$newEditor; Quick="RM" }
)
```

14.5.11.7 Related Cmdlets

- *Get-KFPermission*
- *Reset-KFPermissionInheritance*
- *Set-KFPermission*
- *Stop-KFPermissionInheritance*
- *Update-KFPermission*

14.5.12 Clear-KFDataTime

14.5.12.1 Syntax

```
Clear-KFDateTime
  [-ThrowErrors[:$true]]
```

14.5.12.2 Support

>= *kenaflow* 2.0.0

14.5.12.3 Return

Nothing.

14.5.12.4 Description

This is a tool cmdlet for debugging.

Together with *Get-KFDateTime* and *Set-KFDateTime* this cmdlet can be used to simulate time differences in a debugging context. First you set a time difference with `Set-KFDateTime` that is relative to the current time. When you get the time using `Get-KFDateTime` the difference is added or substracted from the current time. In combination with *Get-KFInDebug* you can simulate a workflow run "later" or "earlier", e.g. if you check time stamps against the current time.

With `Clear-KFDataTime` you can clear the previously set time difference.

14.5.12.5 Parameters

`[-ThrowErrors[:$true]]`

> This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.12.6 Example

```
Clear-KFDataTime
```

14.5.12.7 Related Cmdlets

- *Get-KFDateTime*
- *Set-KFDateTime*
- *Get-KFInDebug*

14.5.13 ConvertFrom-KFISODateTime

14.5.13.1 Syntax

```
ConvertFrom-KFISODateTime
  [-Dt] <string>
  [-ThrowErrors[:$true]]
```

14.5.13.2 Support

>= *kenaflow* 1.0.0

14.5.13.3 Return

SystemDateTime object.

14.5.13.4 Description

The command converts a SharePoint-style representation of dates and times into a PowerShell date object.

14.5.13.5 Parameters

`[-Dt] <string>`

Datetime value as string.

`[-ThrowErrors[:$true]]`

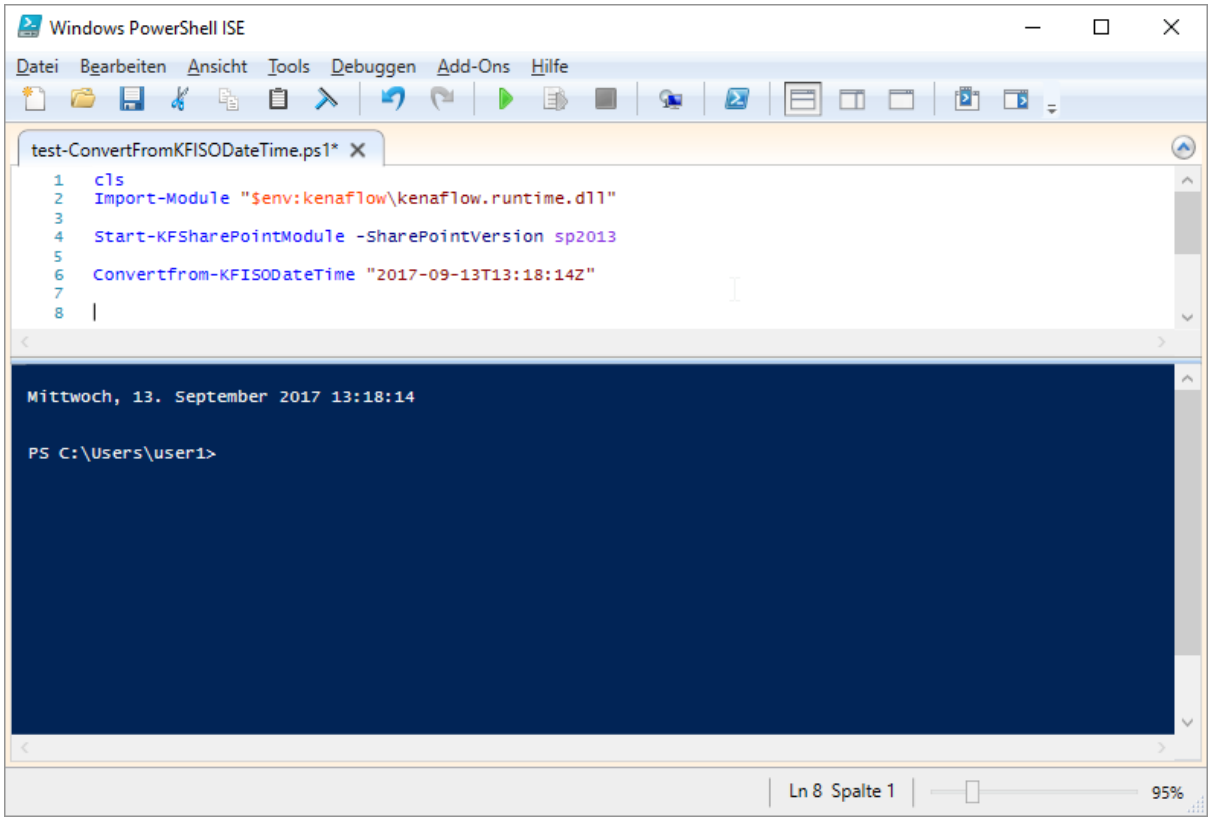
This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.13.6 Example

```
Cls; Import-Module "c:\Program Files\kenaflow\kenaflow.runtime.dll"

Start-KFSharePointModule -SharePointVersion sp2013

ConvertFrom-KFISODateTime "2017-09-13T13:18:14Z"
```



14.5.13.7 Related Cmdlets

- *ConvertTo-KFISODateTime*

14.5.14 ConvertFrom-KFJSON

14.5.14.1 Syntax

```
ConvertFrom-KFJSON
  [-JSON] <string>
  [-ThrowErrors[:$true]]
```

14.5.14.2 Support

>= *kenaflow* 1.0.0

14.5.14.3 Return

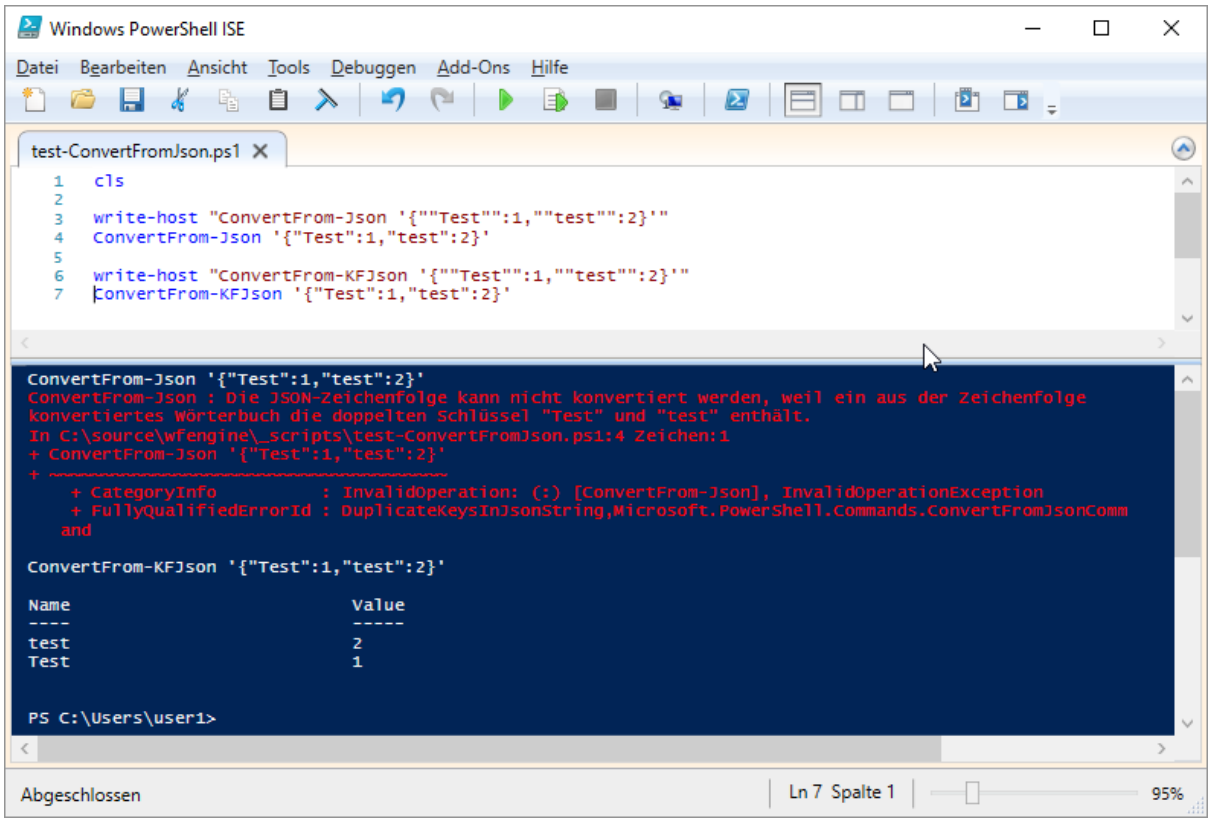
PowerShell Hashtable object.

14.5.14.4 Description

The default PowerShell command `ConvertFrom-JSON` returns JSON serialized objects as PSObject objects (`System.Management.Automation.PSObject`) with properties. This is sometimes not sufficient, because it will fail if the JSON string contains for example two properties with the same name but different cases: `{"Test":"kenaflow","test":"cool"}`.

Our cmdlet converts the JSON to a PowerShell Hashtable. It can handle keys with different upper and lower case.

The following example shows the difference between the default Cmdlet and the *kenaflow* Cmdlet:



14.5.14.5 Parameters

`[-JSON] <string>`

The JSON serialized object to be deserialized.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.14.6 Example

```
ConvertFrom-KFJSON -JSON '{"Test":"kenaflow","test":"cool"}'
```

14.5.15 ConvertoTo-KFCAML

14.5.15.1 Syntax

```
Get-KFNormalizedCaml
  [-Query] <string>
  [-Config <hashtable>]
  [-ThrowErrors[:$true]]
```

14.5.15.2 Support

>= *kenaflow* 1.0.0

14.5.15.3 Return

String.

14.5.15.4 Description

The cmdlet is only available for SharePoint workflows!

This Cmdlet converts queries in *kenaflow*'s own query language into common CAML.

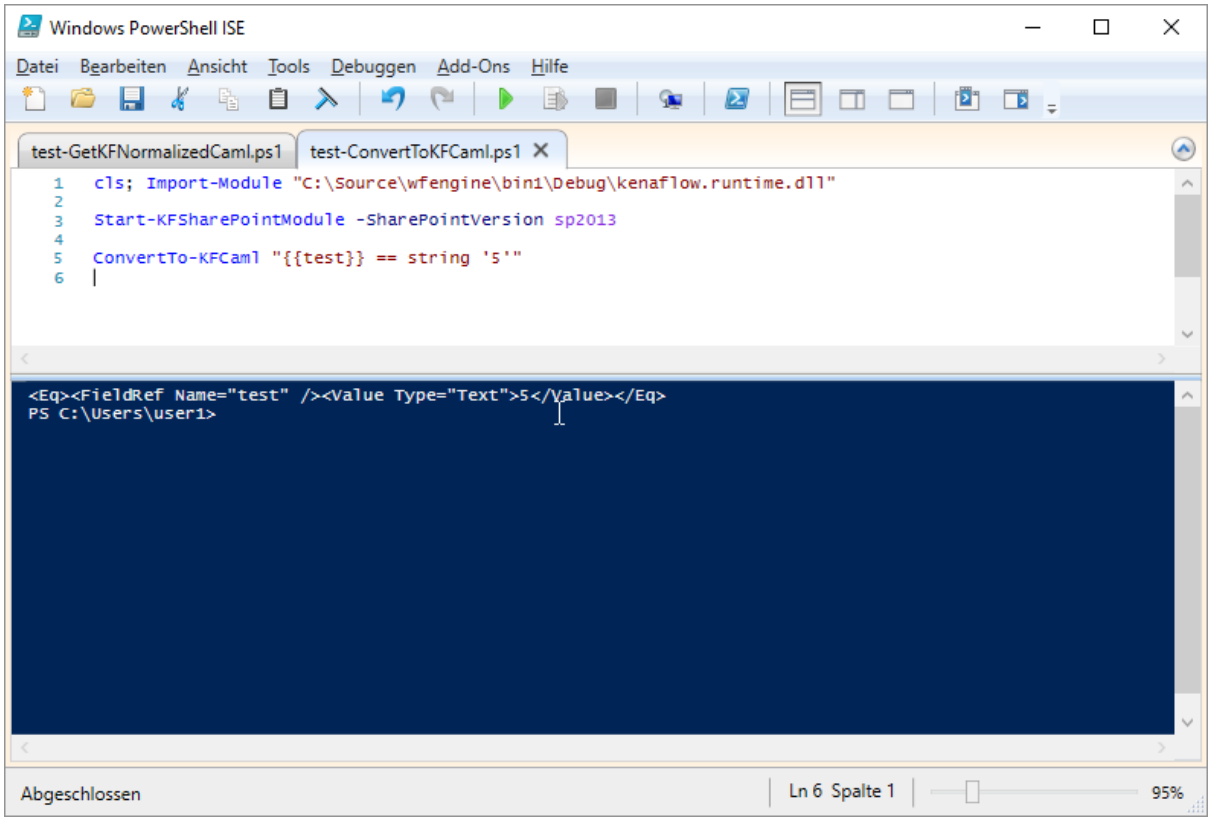
Please read this aricle get get familar with *kenaflow Queries*.

This cmdlet can be used to test the *kenaflow* query processing mechanism to inspect the CAML query.

```
Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll"

Start-KFSharePointModule -SharePointVersion sp2013

ConvertTo-KFCaml "{{test}} == string '5' & {{test2}} != null"
```



The query can contain placeholders of type "config" surronded with `[[` and `]]`. At runtime they are replaced by values from the config entries from the SharePoint config list if the config list is configured.

In case of testing a CAML query outside of an automated workflow run you can use out cmdlet `ConvertTo-KFCaml` with parameter `-Config <hashtable>`:

```
Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll"

Start-KFSharePointModule -SharePointVersion sp2013

$config = @{ "ourValue" = "5" }

ConvertTo-KFCaml "{{test}} == string [[ourValue]] & {{test2}} != null"
```

14.5.15.5 Parameters

`[-Query] <string>`

The *kenaflow* query string.

`[-Config <hashtable>]`

A PowerShell Hashtable object that contains config values for replacement.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.15.6 Example

Examples are in the description above.

14.5.16 ConvertTo-KFISODateTime

14.5.16.1 Syntax

```
ConvertFrom-KFISODateTime
  [-Dt] <datetime>
  [-ThrowErrors[:$true]]
```

14.5.16.2 Support

>= *kenaflow* 1.0.0

14.5.16.3 Return

String.

14.5.16.4 Description

This cmdlet converts PowerShell date objects to the ISO representation that can be used with SharePoint.

14.5.16.5 Parameters

`[-Dt] <string>`

Datetime value as string.

`[-ThrowErrors[:$true]]`

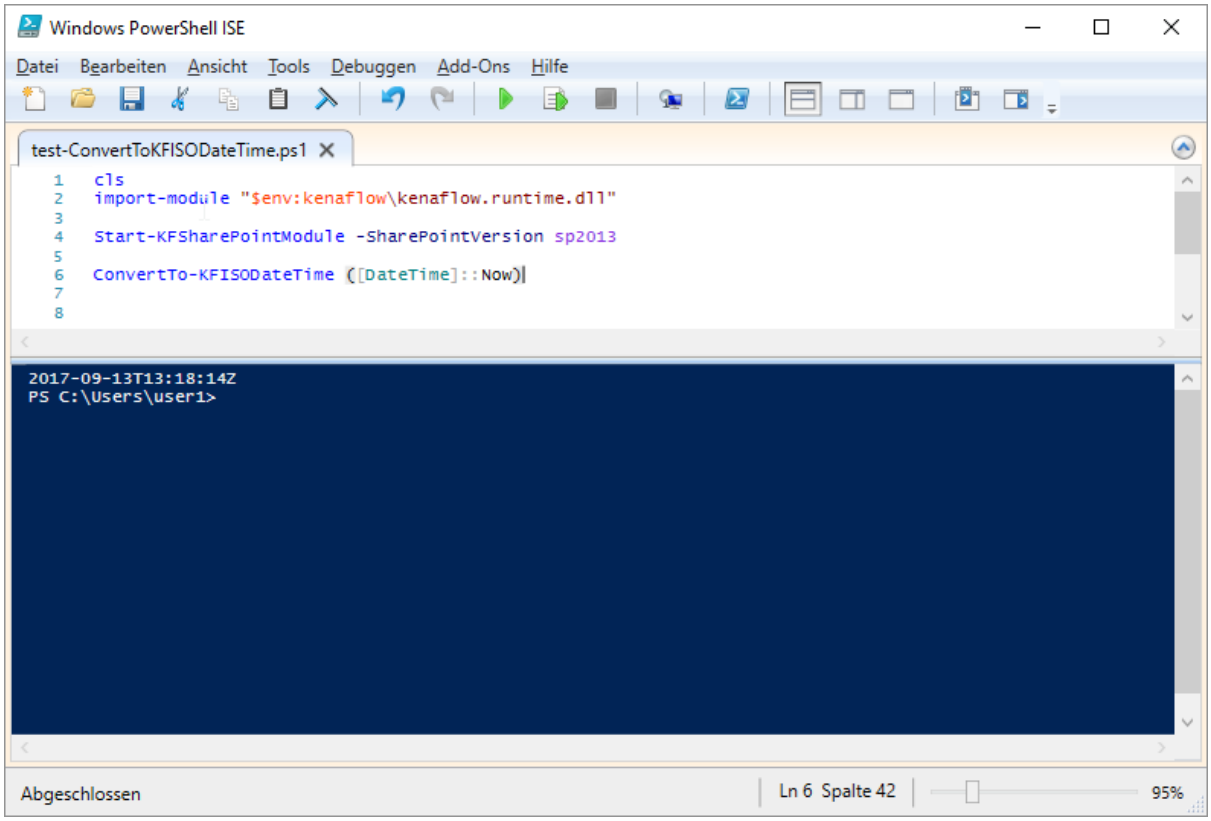
This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.16.6 Example

```
Cls; Import-Module "c:\Program Files\kenaflow\kenaflow.runtime.dll"

Start-KFSharePointModule -SharePointVersion sp2013

ConvertTo-KFISODateTime ([DateTime]::Now)
```



14.5.16.7 Related Cmdlets

- *ConvertFrom-KFISODateTime*

14.5.17 Get-KFBarcode

! available: *kenaflow* 2.0.28 and above.

14.5.17.1 Syntax

```
Get-KFBarcode
  -InputStream <stream>
  [-Multiple[:$false]]
  [-TryHarder[:$false]]
  [-ThrowErrors[:$true]]
```

```
Get-KFBarcode
  -InputBase64 <string>
  [-Multiple[:$false]]
  [-TryHarder[:$false]]
  [-ThrowErrors[:$true]]
```

14.5.17.2 Support

>= *kenaflow* 2.0.28

14.5.17.3 Return

Barcode object.

14.5.17.4 Description

We use the open source library ZXING.NET to read barcodes from images.

14.5.17.5 Parameters

`-InputStream <stream>`

The image as System.IO.Stream.

`-InputBase64 <string>`

The image as Base64 string.

`[-Multiple[:$false]]`

Set this parameter if the image contains multiple barcodes.

`[-TryHarder[:$false]]`

Set this parameter to force the algorithm to "try harder" to read the barcode.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.17.6 Related Cmdlets

- *New-KFBarcode*

14.5.18 Get-KFCron

14.5.18.1 Syntax

```
Get-KFCron
  [-Pattern] <string> ("* * * * *")
  [-Dt] <datetime> (System.DateTime.Now)
  [-ThrowErrors[:$true]]
```

14.5.18.2 Support

>= kenaflow 2.0.0

14.5.18.3 Return

Object of type kenaflow.CronDateTime.

14.5.18.4 Description

"Cron" is a job scheduling pattern known from Un*x and Linux.

With this command you can calculate the next valid date and time for a given start date (server local time) and a a CRON pattern.

You can specify a CRON pattern for workflow execution by setting the property `CRON` in `_wfconfig.ps1`.

The CRON syntax can be found here:

https://en.wikipedia.org/wiki/Cron

We will not describe the syntax here.

kenaflow supports:

- all match `*` ("all")
- single numbers
- lists, like `1,3,5,20,25,45`
- ranges, like `1-30`
- frequency, like `5/10`

The return object of type kenaflow.CronDateTime looks like this:

```
public class CronDateTime
{

    public DateTime Next();

    public void Reset();

    public DateTime Base {get;}

    public DateTime Current {get;}
}
```

You can get such an object with `Get-KFCron`. Its based on a given System.DateTime and offers method `Next()` to get the next valid DateTime matching the pattern.

When calling `Next()` the new value is the basis for the next call to `Next()`.

With `Reset()` you can reset the sequence.

With `Base` you get the initial `DateTime` value.

With `Current` you get the current `DateTime` value from the initialization or the last call of `Next()`.

14.5.18.5 Parameters

`[-Pattern] <string> ("* * * * *")`

The CRON pattern.

`[-Dt] <datetime> (System.DateTime.Now)`

The DateTime as start reference.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.18.6 Example

```
Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll"
$dt = [DateTime]::Now.AddDays(10)
$cronDt = Get-KFCron -Dt $dt -Pattern "5 6 * * *"
$cron.Next()
```

14.5.18.7 Related Cmdlets

- *Get-KFCronUtc*

14.5.19 Get-KFCronUtc

14.5.19.1 Syntax

```
Get-KFCronUtc
  [-Pattern] <string> ("* * * * *")
  [-Dt] <datetime> (System.DateTime.UtcNow)
  [-ThrowErrors[:$true]]
```

14.5.19.2 Support

>= kenaflow 3.0.13

14.5.19.3 Return

Object of type kenaflow.CronDateTime.

14.5.19.4 Description

"Cron" is a job scheduling pattern known from Un*x and Linux.

With this command you can calculate the next valid date and time for a given start date (UTC) and a a CRON pattern.

The CRON syntax can be found here:

https://en.wikipedia.org/wiki/Cron

We will not describe the syntax here.

kenaflow supports:

- all match * ("all")
- single numbers
- lists, like `1,3,5,20,25,45`
- ranges, like `1-30`
- frequency, like `5/10`

The return object of type kenaflow.CronDateTime looks like this:

```
public class CronDateTime
{

    public DateTime Next();

    public void Reset();

    public DateTime Base {get;}

    public DateTime Current {get;}
}
```

You can get such an object with `Get-KFCronUtc`. Its based on a given System.DateTime and offers method `Next()` to get the next valid DateTime matching the pattern.

When calling `Next()` the new value is the basis for the next call to `Next()`.

With `Reset()` you can reset the sequence.

With `Base` you get the initial `DateTime` value.

With `Current` you get the current `DateTime` value from the initialization or the last call of `Next()`.

14.5.19.5 Parameters

`[-Pattern] <string> ("* * * * *")`

The CRON pattern.

`[-Dt] <datetime> (System.DateTime.NowUtc)`

The DateTime as start reference.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.19.6 Example

```
Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll"
$dt = [DateTime]::UtcNow.AddDays(10)
$cronDt = Get-KFCronUtc -Dt $dt -Pattern "5 6 * * *"
$cron.Next()
```

14.5.19.7 Related Cmdlets

- *Get-KFCron*

14.5.20 Get-KFData

14.5.20.1 Syntax

```
    Get-KFData
      [-Key] <string>
      [-ThrowErrors[:$true]]
```

14.5.20.2 Support

>= *kenaflow* 1.0.0

14.5.20.3 Return

Object.

14.5.20.4 Description

This cmdlet retrieves the data persisted with *Set-KFData*.

You get a PowerShell object from the deserialized data that is retrieved from the SharePoint workflow *data list*. (The actual name of the list is specified in the workflow configuration file `_wfconfig.ps1`.)

If the requested object does not exist, then `$null` is returned. An exception is not thrown.

14.5.20.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.20.6 Example

```
    Set-KFData -Key "kenaflow" -Data @{"is" = "a workflow engine for SharePoint,␣
↪Email and PowerShell"}
    Write-KFLog (Get-KFData -Key "kenaflow")
    Remove-KFData -Key "kenaflow"
```

14.5.20.7 Related Cmdlets

- *Remove-KFData*
- *Set-KFData*

14.5.21 Get-KFDateTime

14.5.21.1 Syntax

```
Get-KFDateTime
  -Dt <system.datetime>
  -DtUtc <system.datetime>
  [-Real[:$false]]
  [-UTC[:$false]]
  [-DateOnly[:$false]]
  [-AsString[:$false]]
  [-IncludeSeconds[:$false]]
  [-IncludeTime[:$false]]
  [-Format <format-string>]
  [-ThrowErrors[:$false]]
```

14.5.21.2 Support

>= *kenaflow* 1.0.0

14.5.21.3 Return

DateTime or String.

14.5.21.4 Description

This is a tool cmdlet intended to use for debugging and for production.

Together with *Get-KFDateTime* and *Set-KFDateTime* this cmdlet can be used to simulate time differences in a debugging context. First you set a time difference with `Set-KFDateTime` that is relative to the current time. When you get the time using `Get-KFDateTime` the difference is added or substracted from the current time. In combination with *Get-KFInDebug* you can simulate a workflow run "later" or "earlier", e.g. if you check time stamps against the current time.

With `-Dt <datetime>` or `-DtUtc <datetime>` you can specify a custom DateTime. If non is specified `DateTime.Now` is used. If both are specified the value of `-Dt` is used and `-DtUtc` is ignored.

When you specify `-Real` the previously with `Set-KFDateTime` set time difference is **not** used.

When you specify the parameter `-UTC` the value of `-Dt` (or `-DtUtc`) is converted to Utc.

When you specify the parameter `-DateOnly` the time part is removed.

When a time difference was set with `Set-KFDateTime` (and not cleared with `Clear-KFDateTime`) the time difference is added to the DateTime.

When `-AsString` is **not specified** the calculated DateTime is now returned by the cmdlet.

When `-AsString` **is** specified the calculated DateTime is now converted to a string.

If **no** `-Format` is specified following formats are used:

- If `-DateOnly` if specified: `yyyy-MM-dd` is used ***OR***

- If hour, minute and seconds of the calculated DateTime are all "0" and `-IncludeTime` is **not** specified: `yyyy-MM-dd` is used ***OR***
- If `-IncludeTime` is specified but **not** `-IncludeSeconds`: `yyyy-MM-dd HH:mm` is used ***OR***
- If `-IncludeTime` **and** `-IncludeSeconds` are specified: `yyyy-MM-dd HH:mm:ss` is used

Sounds complicated? In SharePoint the "DateOnly" field contains a complete DateTime but with hour == 0, minute == 0 and second == 0.

Therefore when using `-AsString` without other parameters you get only the date part for a DateTime with hour == 0, minute == 0, second == 0. But with hour and minute in the returned string for DateTime values with hour != 0, minute != 0 and/or second != 0.

14.5.21.5 Parameters

`-Dt <system.datetime>`

DateTime to be processed. Default is $null.

`-DtUtc <system.datetime>`

DateTime to be processed. Default is $null.

`[-Real[:$false]]`

If set: continue processing with the real DateTime. Do not add the set time difference.

`[-UTC[:$false]]`

Handle the currently processed DateTime as UTC.

`[-DateOnly[:$false]]`

Continue only with the "date" part of the DateTime.

`[-AsString[:$false]]`

Return the DateTime as string.

`[-IncludeSeconds[:$false]]`

Include Seconds in the returned string. (If `-IncludeTime` is used and `-DateOnly` is not used.)

`[-IncludeTime[:$false]]`

Include the time part of the DateTime in the returned string (if `-DateOnly` is not used). It's used to force also hour == 0, minute == 0 and second == 0 to be included.

`[-Format <format-string>]`

A custom DateTime format. If specified the settings `-DateOnly`, `-IncludeTime` and `-IncludeSeconds` are ignored.

`[-ThrowErrors[:$false]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.21.6 Example

```
Set-KFDateTime -ReferenceDateTime  ([DateTime]"1900-01-01 12:00:00")
Start-Sleep -Seconds 5
Get-KFDateTime -AsString -Format "yyyy-MM-dd HH:mm:ss"
```

This will return `1900-01-01 12:00:05`.

14.5.21.7 Related Cmdlets

- *Set-KFDateTime*
- *Clear-KFDataTime*
- *Get-KFInDebug*

14.5.22 Get-KFDBData

14.5.22.1 Syntax

```
Get-KFDBData
  [-Key] <key>
  [-RelatedToWorkflow[:$true]]
  [-RelatedToItem[:$true]]
  [-ThrowErrors[:$true]]
```

14.5.22.2 Support

>= *kenaflow* 1.0.0

14.5.22.3 Return

Object.

14.5.22.4 Description

This cmdlet retrieves the data persisted with *Set-KFDBData*.

You get a PowerShell object from the deserialized data that is retrieved from the *kenaflow* SQLite database.

If the requested object does not exist, then `$null` is returned. An exception is not thrown.

We have described this in article *Persisting Data*, section "*kenaflow* SQLite Persistence".

By default `-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` are used - alone or in combination - to automatically add prefix(es) to your key.

14.5.22.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.22.6 Example

```
    Set-KFDBData -Key "kenaflow" -Data @{"is" = "a workflow engine for SharePoint,␣
↪Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:$false
    Write-KFLog (Get-KFDBData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false)
    Remove-KFDBData -Key "kenaflow" -RelatedToItem:$false -RelatedToWorkflow:$false
```

14.5.22.7 Related Cmdlets

- *Set-KFDBData*
- *Remove-KFDBData*

14.5.23 Get-KFHash

14.5.23.1 Syntax

```
    Get-KFHash
      -String <string>
      [-ThrowErrors[:$true]]
```

```
    Get-KFHash
      -Object <object>
      [-ThrowErrors[:$true]]
```

14.5.23.2 Support

>= *kenaflow* 2.0.7

14.5.23.3 Return

Base64 encoded string.

14.5.23.4 Description

This cmdlet creates a SHA512 hash as Base64 encoded string from a String or an object.

If an object is supplied it is serialized as PowerShell CliXML before hashing it.

14.5.23.5 Parameters

`-String <string>`

The string that has to be hashed.

`-Object <object>`

The object that has to be hashed.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.23.6 Example

```
Get-KFHash -Object @{Hello="kenaflow"}
```

14.5.24 Get-KFInDebug

14.5.24.1 Syntax

```
Get-KFInDebug
  -EvalAllStates
  [-ThrowErrors[:$true]]
```

```
Get-KFInDebug
  -ScriptDebugState
  [-ThrowErrors[:$true]]
```

```
Get-KFInDebug
  -WorkflowDebugState
  [-ThrowErrors[:$true]]
```

```
Get-KFInDebug
  -WorkflowStateDebugState
  [-ThrowErrors[:$true]]
```

14.5.24.2 Support

>= *kenaflow* 2.0.3

14.5.24.3 Return

Boolean.

`$true` means that the workflow is "in debug mode"

`$false` means the workflow is running in automatic mode.

14.5.24.4 Description

This Cmdlet returns `$true` when the workflow script is in debug mode. Otherwise `$false`.

This allows code to be built into workflow scripts, which is executed only in the debug case.

14.5.24.5 Parameters

`-EvalAllStates`

> This evaluates all three possible sources for the debug flag. If at least one of the states yields `$true` then the Cmdlet also returns `$true`.

`-ScriptDebugState`

> With this switch parameter the cmdlet will return `$true` only then debugging a script in an PowerShell debugger like PowerShell ISE or Visual Studio Code.

`-WorkflowDebugState`

> In `_wfconfig.ps1` the parameter `Debug = $true` can be specified to mark the top-level workflow as a "development version".

`-WorkflowStateDebugState`

> In `_wfconfig.ps1` each status of a Statemachine Workflow can be marked separately with the parameter `Debug = $true`.

`[-ThrowErrors[:$true]]`

> This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.24.6 Examples

```
    $timestamp = $item["Modified"]
    if( (Get-KFInDebug) ) {
            $timestamp = $timestamp.AddDays(1)
    }

    if( $timestamp -lt (Get-KFDateTime -Utc) ) {
            Send-KFMail -Subject "Reminder" -Body "Please do something" -To "{
↪{Author}};{{Editor}}"
    }
```

14.5.25 Get-KFItemData

14.5.25.1 Syntax

```
Get-KFItemData
  [[-Key] <string>]
  [-Item <list-item-object>]
  [-ThrowErrors[:$true]]
```

14.5.25.2 Support

>= *kenaflow* 1.0.0

14.5.25.3 Return

Object.

14.5.25.4 Description

!! This is only available in SharePoint *list* or *state machine* workflows!

This cmdlet retrieves the data previously persisted with *Set-KFItemData*.

If the requested object does not exist, then `$null` is returned. There is no error.

14.5.25.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-Item <list-item-object>]`

The SharePoint list item in whose context the data is serialized.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.25.6 Example

```
    $item = Get-PnPListItem -List "Custom List" -Id 1
    Set-KFItemData -Item $item -Key "kenaflow" -Data @{"is" = "a workflow engine␣
↪for SharePoint, Email and PowerShell"}
    Write-KFLog (Get-KFItemData -Item $item -Key "kenaflow")
    Remove-KFItemData -Item $item -Key "kenaflow"
```

14.5.25.7 Related Cmdlets

- *Set-KFItemData*
- *Remove-KFItemData*

14.5.26 Get-KFLastResult

14.5.26.1 Syntax

```
Get-KFLastResult
  [-ThrowErrors[:$true]]
```

14.5.26.2 Support

>= *kenaflow* 2.0.0

14.5.26.3 Return

Nothing.

14.5.26.4 Description

Many cmdlets from *kenaflow* return values. These values are output to the PowerShell pipeline. These values can subsequently be queried with the `Get-KFLastResult` cmdlet.

Furthermore, this cmdlet is used to query error messages from *kenaflow* cmdlets if a *kenaflow* cmdlet terminated with an error *and* the error was not outputted to the PowerShell runtime using the `ThrowErrors:$false` option.

For example, the following script can be used to check the existence of a user:

```
Get-KFUserInfo -UserName 'kenaflow' -ThrowErrors:$false
$e = Get-KFLastResult
if( $e -is [System.Exception]) {
Write-Host "User does not exist"
}
```

If the user "kenaflow" does not exist, then the Cmdlet `Get-KFUserInfo` does not throw an error which causes the script to exit, but the exception is caught and can be retrieved by `Get-KFLastResult`. If the user "kenaflow" would exist in this example, the variable `$e` contains the user object.

14.5.26.5 Parameters

`[-ThrowErrors[:$true]]`

> This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.26.6 Example

It's in the description above.

14.5.27 Get-KFLink

14.5.27.1 Syntax

```
Get-KFLink
  [-Item <list-item> (the currently processed one)]
  [-WorkflowId (the current workflows id)]
  [-Anonymous[:$false]]
  [-ValidUntil <datetime>]
  [-WithAnswer[:$true]]
  [-Metadata <hashtable> ($null)]
  [-ThrowErrors[:$true]]
```

14.5.27.2 Support

>= *kenaflow* 1.0.0

14.5.27.3 Return

A string that represents the "rich link".

14.5.27.4 Description

This cmdlet can be used to create "rich links". If the user clicks the generated link a "link remote event" is fired. The browser opens and the users gets a generic answer or an answer from the workflow.

The cmdlet is only available in SharePoint workflows!

Please read this article for details: *Links*.

14.5.27.5 Parameters

`[-Item <list-item> (the currently processed one)]`

> You can specify a different list item for which you want to create a rich link. Default is the currently processed item.

`[-WorkflowId (the current workflows id)]`

> You can specify the ID of a workflow that should process the link remote event.

`[-Anonymous[:$false]]`

> if specified the link will point to a http endpoint that does not require authentication. The user is unknown to the workflow in this case.

`[-ValidUntil <datetime>]`

> You can specify a "lifetime" for the link. If it is expired the user gets a generic answer in the browser when he/she clicks the link.

`[-WithAnswer[:$true]]`

> If `$true` (default) the browser opens and waits for the answer of the workflow. This is required to be a HTML page.

`[-Metadata <hashtable> ($null)]`

You can transfer data to the workflow. This data is serialized.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.27.6 Example

```
    Get-KFLink -Metadata @{Action="accepted"} -ValidUntil ([DateTime]::Now.
↪AddDays(1))
```

14.5.28 Get-KFNormalizedCAML

14.5.28.1 Syntax

```
    Get-KFNormalizedCaml
      [-Query] <string>
      [-ThrowErrors[:$true]]
```

14.5.28.2 Support

>= *kenaflow* 1.0.0

14.5.28.3 Return

String.

14.5.28.4 Description

If you want to work directly with CAML - the SharePoint query language - then you need to keep in mind that the `And` and `Or` tags must each have exactly two suitable sub-objects.

Not valid would be:

```
    <Where><And><Eq>...</Eq></And></Where>
```

...because here is just one `Eq` under ´And´.

Also not valid is:

```
    <Where><Or><Eq>...</Eq><Gt>...</Gt><Lt>...</Lt></Or></Where>
```

...because here the `Or`-Tag has three subnodes (`Eq`, `Gt`, `Lt`).

This kind of problems occur, if CAML is built dynamically.

14.5.28.5 Parameters

`[-Query] <string>`

This is the CAML string to be corrected.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

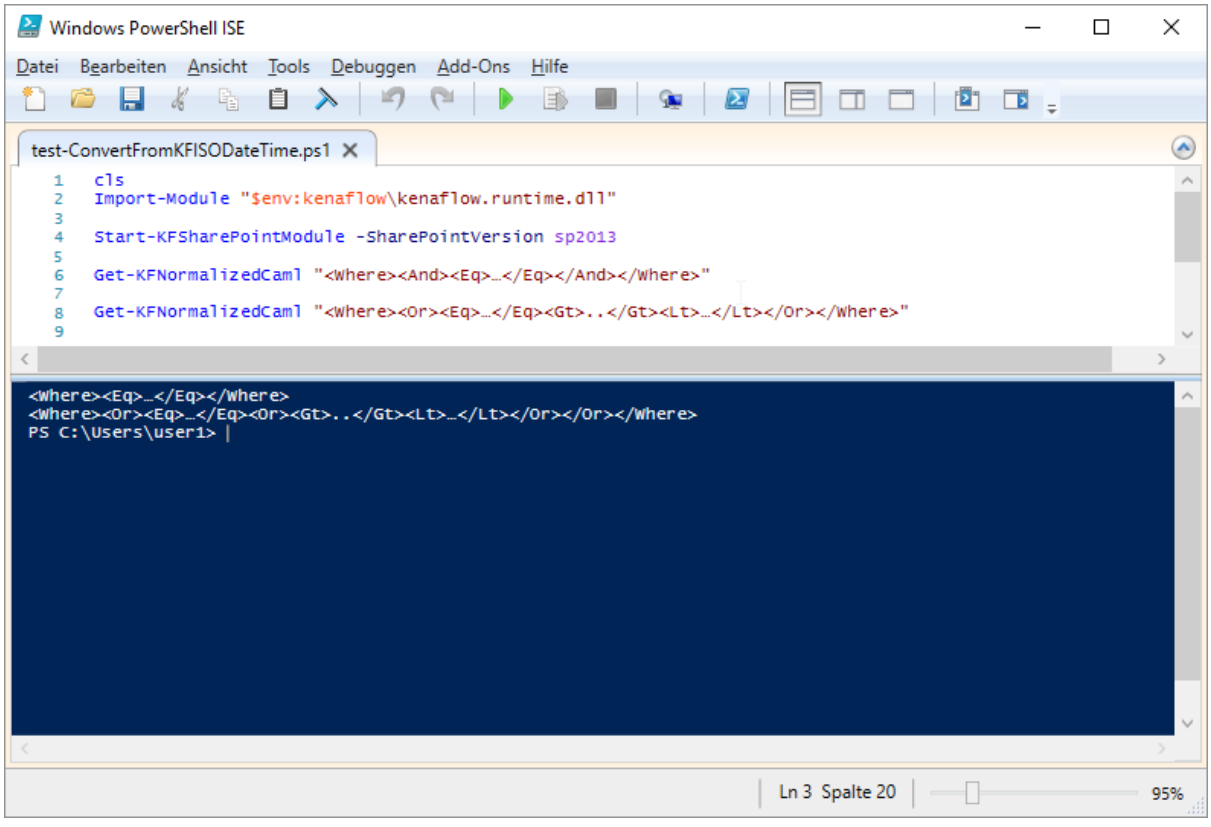
14.5.28.6 Example

```
    Cls; Import-Module "C:\Program Files\kenaflow.runtime.dll"

    Start-KFSharePointModule -SharePointVersion sp2013

    Get-KFNormalizedCaml -Query "<Where><And><Eq>...</Eq></And></Where>"

    Get-KFNormalizedCaml "<Where><Or><Eq>..</Eq><Gt>..</Gt><Lt>..</Lt></Or></Where>
↪"
```



14.5.29 Get-KFObject

14.5.29.1 Syntax

```
Get-KFObject
  -Name <string>
  [-ThrowErrors[:$true]]
```

14.5.29.2 Support

>= *kenaflow* 4.0.11

14.5.29.3 Return

14.5.29.4 Description

This cmdlet can be used to read data from the in-memory-cache previously written by *Set-KFObject*.

14.5.29.5 Parameters

`[-Name] <string>`

The key that is used to write data

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.29.6 Example

The `proProcessingScript` specified in `_wfconfig.ps1` could include the following line of Powershell script code:

```
Set-KFObject -Name "ListOfDataForExcelExport" -Object @()
```

The SharePoint workflow script could include this script code:

```
$d = Get-KFObject -Name "ListOfDataForExcelExport"
$d += @{Title = $item["Title"]; Id = $item.Id; Author = $item["Author"].
↪LookupValue; Editor = $item["Editor"].LookupValue}
Set-KFObject -Name "ListOfDataForExcelExport" -Object $d
```

This creates an empty list in memory.

The `postProcessingScript` specified in `_wfconfig.ps1` could include the following line of Powershell script code:

```
$data = Get-KFObject -Name "ListOfDataForExcelExport"
New-KFExcelFile -Input $data -SheetName "Output" -AsBase64 -Fields @("Id",
↪"Title", "Author", "Editor")

Remove-KFObject -Name "ListOfDataForExcelExport" # This is not required␣
↪because at the end of a workflow processing the in-memory-object is cleared!
```

14.5.29.7 Related Cmdlets

- *Set-KFObject*
- *Remove-KFObject*

14.5.30 Get-KFPermission

14.5.30.1 Syntax

```
Get-KFPermission
  [-Item <list-item>]
  [-ThrowErrors[:$true]]
```

14.5.30.2 Support

>= *kenaflow* 1.0.0

14.5.30.3 Return

List of principals and their permission sets on the list item.

14.5.30.4 Description

Use this cmdlet to read the permissions of the current list item or the list item specified with `-Item`.

The result object is a Hashtable of all principals (SharePoint users and SharePoint groups) that have permissions on the list item. For each principal the permission sets are returned.

14.5.30.5 Parameters

`[-Item <list-item>]`

The list item to be processed.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.30.6 Examples

```
Get-KFPermission
```

14.5.30.7 Related Cmdlets

- *Add-KFPermission*
- *Reset-KFPermissionInheritance*
- *Set-KFPermission*
- *Stop-KFPermissionInheritance*
- *Update-KFPermission*

14.5.31 Get-KFPrettyXML

14.5.31.1 Syntax

```
Get-KFPrettyXML
  [-xml] <string>
  [-ThrowErrors[:$true]]
```

14.5.31.2 Support

>= *kenaflow* 1.0.0

14.5.31.3 Return

String.

14.5.31.4 Description

With this cmdlet, poorly readable XML can be transformed into a hierarchical, readable form.

14.5.31.5 Parameters

`[-xml] <string>`

The XML string to be formatted.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.31.6 Example

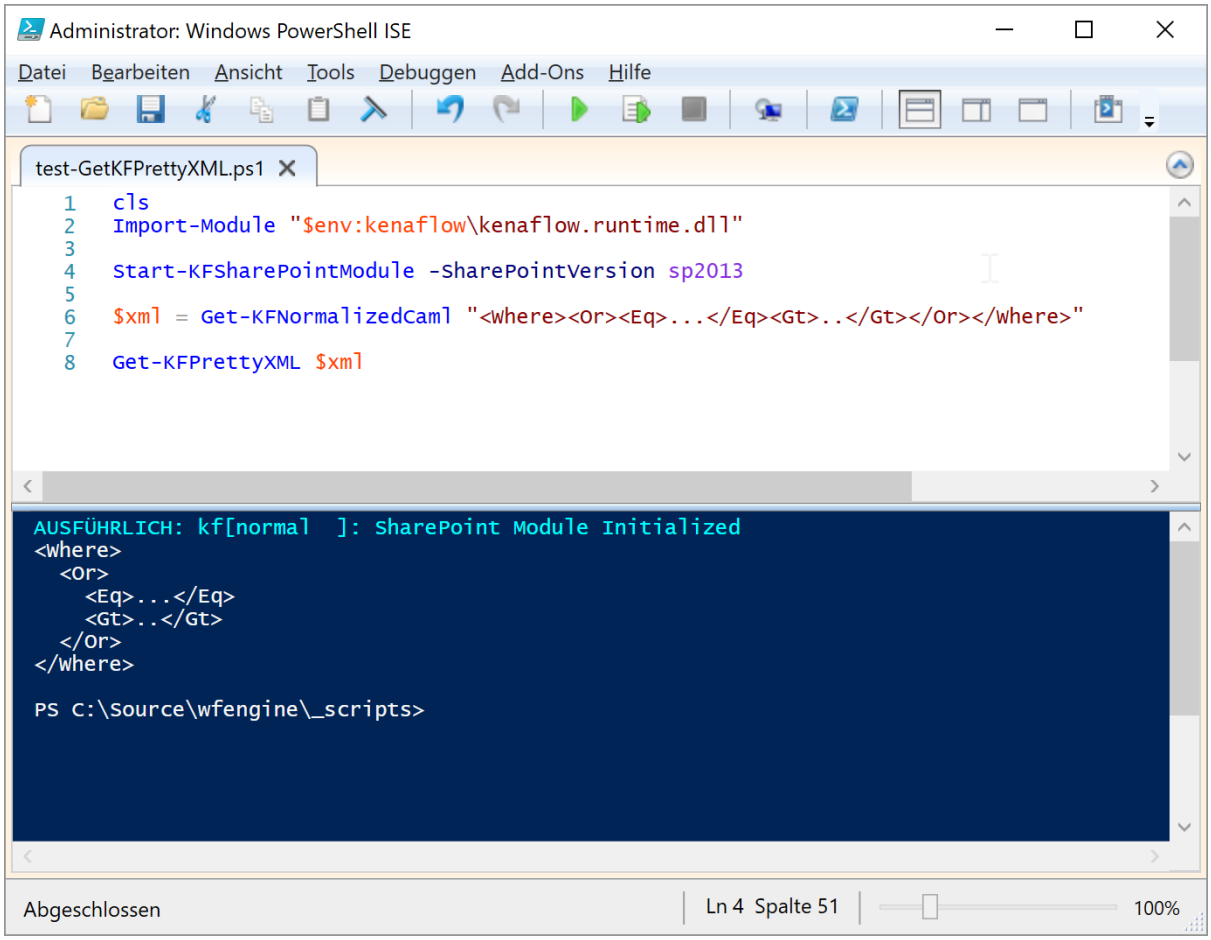
```
Cls; Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll"

Start-KFSharePointModule -SharePointVersion sp2013

$xml = Get-KFNormalizedCaml "<Where><Or><Eq>...</Eq><Gt>..</Gt></Or></Where>"

Get-KFPrettyXML $xml
```

The String `<Where><Or><Eq>...</Eq><Gt>..</Gt></Or></Where>` is reformatted and returned.



14.5.32 Get-KFScriptRunTimeout

14.5.32.1 Syntax

```
Get-KFScriptRunTimeout
    -Remaining
```

```
Get-KFScriptRunTimeout
        -TotalScriptTimeout
```

14.5.32.2 Support

>= *kenaflow* 2.0.18

14.5.32.3 Description

Workflow scripts are executed inside PowerShell. As in every programming language it is possible in PowerShell to create loops. It is not possible for *kenaflow* to check the "code quality" of a workflow script. If you (accidentally) create an endless loop *kenaflow* would not be able to stop the script. Therefore it has an security feature that forces scripts to stop if they run too long.

The initial setting for maximum script runtime is set in Global Configuration setting `` or in the workflow configuration ***missing***.

If a workflow knows that it needs more time the execution timeout can be set inside the workflow script during execution with `Set-KFScriptRunTimeout`

With the cmdlet `Get-KFScriptRunTimeout` you can get the remaining amount of seconds or the total amout of seconds before the script times out.

14.5.32.4 Parameter

`-Remaining`

Get the remaining amount of seconds before the script times out.

`-TotalScriptTimeout`

Get the total amount of seconds before the script times out.

14.5.32.5 Examples

```
***missing***
```

14.5.33 Get-KFSessionStorageData

14.5.33.1 Syntax

```
Get-KFSessionStorageData
  [-Key] <key>
  [-RelatedToWorkflow[:$true]]
  [-RelatedToItem[:$true]]
  [-Binary:[$false]]
  [-AdditionalPassPhrase <passphrase>]
  [-ThrowErrors[:$true]]
```

14.5.33.2 Support

>= *kenaflow* 3.0.13

14.5.33.3 Return

Object or `System.Byte[]`

14.5.33.4 Description

This cmdlet retrieves the data persisted with *`Set-KFSessionStorageData`*.

The data is read from the session storage of *kenaflow*. Please read the article about *Storage*.

When writing data with `Set-KFSessionStorageData` the workflow developer decides whether the data is encrypted or not. When reading the data with `Get-KFSessionStorageData` *kenaflow* reads the from the storage and decrypts it if needed.

If the data was written using an `AdditionalPassphrase` this must be specified for proper decryption.

If the requested object does not exist, then `$null` is returned. An exception is not thrown.

`-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` can be used - alone or in combination - to automatically add prefix(es) to your key.

14.5.33.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-Binary:[$false]]`

If NOT specified the cmdlet will return a deserialized object. This requires that data for the given `Key` was written NOT with `-Binary`. If specified the cmdlet will return an byte array (`System.Byte[]`). This requires that data for the given `Key` was written with `-Binary`.

`[-AdditionalPassPhrase <passphrase>]`

Workflow developers can specify an additional encryption passphrase with `Set-KFStorageData`. This needs to be specified for retrieving the data!

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.33.6 Example

This writes and reads serialized data to the session storage.

```
    Set-KFSessionStorageData -Key "kenaflow" -Data @{"is" = "a workflow engine for␣
↪SharePoint, Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:
↪$false -AdditionalPassphare "cool"
    Write-KFLog (Get-KFSessionStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false -AdditionalPassphare "cool")
    Remove-KFSessionStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false
```

14.5.33.7 Related Cmdlets

- *Get-KFSessionStorageData*
- *Set-KFSessionStorageData*
- *Remove-KFSessionStorageData*

14.5.34 Get-KFSPDatetimeLocal

14.5.34.1 Syntax

```
    Get-KFSPDatetimeLocal
      [-Item <list-item>]
      -Field <internal-name-of-field>
      [-ThrowErrors[:$true]]

    Get-KFSPDatetimeLocal
      [-Item <list-item>]
      -DateTimeUtc <datetime-in-utc>
      [-ThrowErrors[:$true]]
```

14.5.34.2 Support

>= *kenaflow* 4.0.12

14.5.34.3 Return

A localized DateTime value based on the SharePoint web regional setting.

14.5.34.4 Description

This cmdlet can be used to convert the UTC datetime value in a SharePoint list item field to the localized DateTime value as it was entered or is viewed by the user.

14.5.34.5 Parameters

`[-Item <list-item>]`

You can specify a different list item for which you want to create a rich link. Default is the currently processed item.

`-Field <internal-name-of-field>`

Internal name of a field of the current or specified list item.

`[-Anonymous[:$false]]`

if specified the link will point to a http endpoint that does not require authentication. The user is unknown to the workflow in this case.

`-DateTimeUtc <datetime-in-utc>`

Instead of specifying a field you can specify a DateTime value that will be converted to a localized DateTime value.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.34.6 Example

```
$originalValue = (Get-KFSPDatetimeLocal -Field "SomeDateTimeField")

$newValue = $originalValue.addHours(1)

Set-KFSPDatetimeLocal -Field "SomeDateTimeField" -DateTimeLocal $newValue
$item.Update()

Invoke-PnpQuery
```

14.5.34.7 Related Cmdlets

- *Set-KFSPDatetimeLocal*

14.5.35 Get-KFStorageData

14.5.35.1 Syntax

```
Get-KFStorageData
  [-Key] <key>
  [-RelatedToWorkflow[:$true]]
  [-RelatedToItem[:$true]]
  [-Binary:[$false]]
  [-AdditionalPassPhrase <passphrase>]
  [-ThrowErrors[:$true]]
```

14.5.35.2 Support

>= *kenaflow* 3.0.13

14.5.35.3 Return

Object or `System.Byte[]`

14.5.35.4 Description

This cmdlet retrieves the data persisted with *`Set-KFStorageData`*.

The data is read from the permanent storage of *kenaflow*. Please read the article about *Storage*.

When writing data with `Set-KFStorageData` the workflow developer decides whether the data is encrypted or not. When reading the data with `Get-KFStorageData` *kenaflow* reads the from the storage and decrypts it if needed.

If the data was written using an `AdditionalPassphrase` this must be specified for proper decryption.

If the requested object does not exist, then `$null` is returned. An exception is not thrown.

`-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` can be used - alone or in combination - to automatically add prefix(es) to your key.

14.5.35.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-Binary:[$false]]`

If NOT specified the cmdlet will return a deserialized object. This requires that data for the given `Key` was written NOT with `-Binary`. If specified the cmdlet will return an byte array (`System.Byte[]`). This requires that data for the given `Key` was written with `-Binary`.

`[-AdditionalPassPhrase <passphrase>]`

Workflow developers can specify an additional encryption passphrase with `Set-KFStorageData`. This needs to be specified for retrieving the data!

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.35.6 Example

This writes and reads serialized data to the permanent storage.

```
    Set-KFStorageData -Key "kenaflow" -Data @{"is" = "a workflow engine for␣
↪SharePoint, Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:
↪$false -AdditionalPassphare "cool"
    Write-KFLog (Get-KFStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false -AdditionalPassphare "cool")
    Remove-KFStorageData -Key "kenaflow" -RelatedToItem:$false -RelatedToWorkflow:
↪$false
```

14.5.35.7 Related Cmdlets

- *Set-KFStorageData*
- *Remove-KFStorageData*

14.5.36 Get-KFTest

14.5.36.1 Syntax

```
    Get-KFTest
      [-ThrowErrors[:$true]]
```

14.5.36.2 Support

>= *kenaflow* 1.0.0

14.5.36.3 Return

String. See description below.

14.5.36.4 Description

This cmdlet is used for testing only. It writes a line of text to the PowerShell pipeline:

```
    This is kenaflow <product-version>
```

... where `<product-version>` is you current version number of *kenaflow*.

14.5.36.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.37 Get-KFThrowErrorsPreference

14.5.37.1 Syntax

```
Get-KFThrowErrorsPreference
        [-ThrowErrors[:$true]]
```

14.5.37.2 Support

>= *kenaflow* 2.0.0

14.5.37.3 Return

Boolean.

14.5.37.4 Description

This cmdlet returns the current setting for error handling. This can be set at the level of the workflow script by with *Set-KFThrowErrorsPreference*. The workflows default is read from `_wfconfig.ps1`.

Please read article *Error Handling*.

14.5.37.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.37.6 Example

```
$errorHandling = Get-KFThrowErrorsPreference
if($errorHandling) {
    Write-KFLog "Errors are thrown to PowerShell"
} else {
    Write-KFLog "Errors ere not thrown to PowerShell"
}
```

14.5.37.7 Related Cmdlets

- *Set-KFThrowErrorsPreference*

14.5.38 Get-KFUrl

14.5.38.1 Syntax

```
Get-KFUrl
  -FileOrFolderUrl
  [-Item <item>]
  [-List <list>]
  [-Web <web>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUrl
  -GetItemNewUrl
  [-Item <item>]
  [-List <list>]
  [-Web <web>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUrl
  -GetListDefaultViewUrl
  [-Item <item>]
  [-List <list>]
  [-Web <web>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUrl
  -GetListUrl
  [-Item <item>]
  [-List <list>]
  [-Web <web>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUrl
  -GetWebUrl
  [-Item <item>]
  [-List <list>]
  [-Web <web>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUrl
  -GetItemDisplayUrl
  [-Item <item>]
  [-List <list>]
  [-Web <web>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUrl
  -GetItemEditUrl
  [-Item <item>]
  [-List <list>]
  [-Web <web>]
  [-ThrowErrors[:$true]]
```

14.5.38.2 Support

>= *kenaflow* 1.0.0

14.5.38.3 Return

String.

14.5.38.4 Description

This cmdlet is only available in SharePoint workflows.

This command can be used to output various URLs related to the current workflow, its web, list and currently processed item.

It's also possible to specify custom objects for the used SharePoint web, list or item.

14.5.38.5 Parameters

`-FileOrFolderUrl`

Gets the URL of a file or folder that is represented by the list item. (The current processed one or the on from parameter `-Item`.

`-GetItemNewUrl`

Gets the URL for the "new item" form of the SharePoint list (the workflows list or the one specified in parameter `-List`).

`-GetListDefaultViewUrl`

Gets the default view URL of the SharePoint list (the workflows list or the one specified in parameter `-List`).

`-GetListUrl`

Gets the URL of the SharePoint list (the workflows list or the one specified in parameter `-List`). If opened in browser the default view is shown.

`-GetWebUrl`

Gets the URL of the SharePoint web (the workflows web or the one specified in parameter `-Web`).

`-GetItemDisplayUrl`

Gets the display form URL of the list item. (The current processed one or the on from parameter `-Item`.

`-GetItemEditUrl`

Gets the edit form URL of the list item. (The current processed one or the on from parameter `-Item`.

`[-Item <item>]`

The item to be processed. If not specified the currently processed item is used.

`[-List <list>]`

The list to be processed. If not specified the workflows list item is used.

`[-Web <web>]`

The web to be processed. If not specified the workflows web item is used.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.38.6 Example

```
$item Get-PnPListItem -List "Custom List" -Id 1
Get-KFUrl -GetItemDisplayUrl -Item $item
```

14.5.39 Get-KFUserInfo

14.5.39.1 Syntax

```
Get-KFUserInfo
  -UserField <field-name>
  [-Item <list-item-object>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUserInfo
  -UserName <user-name>
  [-Item <list-item-object>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUserInfo
  -User <User-object>
  [-Item <list-item-object>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUserInfo
  -UserFieldValue <FieldUserValue-object>
  [-Item <list-item-object>]
  [-ThrowErrors[:$true]]
```

```
Get-KFUserInfo
  -UserId <user-id>
  [-Item <list-item-object>]
  [-ThrowErrors[:$true]]
```

14.5.39.2 Support

>= *kenaflow* 1.0

14.5.39.3 Return

PowerShell Hashtable.

14.5.39.4 Description

This cmdlet is only available in SharePoint workflows!

This cmdlet retrieves information about the user from the SharePoint profile store.

The profile store is operated by the "User Profile Service Application". This does not exist in SharePoint Foundation 2013 - that is basically supported by *kenaflow*. Likewise, the cmdlet does not work for this edition of SharePoint.

There are several ways to retrieve the information for a user. Either

- via the username by `-UserName`, e.g. `kenaflow\testuser` (sAMAccountName with domain) or `testuser@kenaflow.com` (userPrincipalName) or

- via a named field with `-UserField` in the currently processed SharePoint list item or the list item specified by `-Item` , e.g. `Author` or `Editor` (both standard fields of SharePoint list items)
- via the user ID with `-UserId`, where a user's ID has nothing to do with the SID in Active Directory, but is specific to each user and each SharePoint Site Collection, e.g. `1` or `15`, or
- via an user object with `-User` or
- via the a given field value with `-UserFieldValue`.

The returned object will then provide various user information, depending on what is stored in the SharePoint profile system or imported from Active Directory and/or other sources.

The return object looks like this:

```
    Name                                Value
    ----                                -----
    SPS-PersonalSiteInstantiationState 2
    SPS-StatusNotes
    ADGuid
    SPS-ObjectExists
    SPS-MasterAccountName
    SPS-PictureExchangeSyncState
    Department
    SPS-SourceObjectDN
    SPS-HashTags
    SPS-Peers
    WorkPhone
    SPS-SavedSID                        System.Byte[]
    PictureUrl
    SPS-HireDate
    SPS-PrivacyPeople                   False
    SPS-Skills
    SPS-MySiteUpgrade
    LastName
    SPS-PrivacyActivity                 0
    SPS-JobTitle
    PublicSiteRedirect
    DirectReports                       {}
    Manager
    SPS-PhoneticDisplayName
    PersonalSpace                       /personal/ingo/
    UserProfile_GUID                    553bd20e-df46-4237-bd2f-b15340518d63
    SPS-SavedAccountName                SHAREPOINT\ingo
    DisplayName                         Ingo
    LatestPost
    SPS-ResourceAccountName
    SPS-PersonalSiteCapabilities        14
    SPS-EmailOptin                      0
    WorkEmail
    SPS-Location
    SPS-ClaimProviderType               Windows
    Assistant
    HomePhone
    IsFollowed                          False
    SPS-School
    SPS-ClaimProviderID                 Windows
    AboutMe
    AccountName                         SHAREPOINT\ingo
    SPS-PhoneticFirstName
    SPS-O15FirstRunExperience
    PersonalUrl                         https://intranet13.sharepoint.farm/personal/
↪ingo/
    CellPhone
```

(continues on next page)

(continued from previous page)

```
    SPS-Responsibility
    ExtendedReports                 {SHAREPOINT\ingo}
    SPS-ProxyAddresses
    SPS-ClaimID
    SPS-PictureTimestamp
    SPS-PastProjects
    SPS-Department
    WebSite
    Title
    FirstName
    PreferredName                   Ingo
    SPS-TimeZone
    SPS-Interests
    UserUrl                         https://intranet13.sharepoint.farm:443/msh/
↪Person.aspx?accountname=SHAREPOINT%5Cingo
    SPS-Birthday
    SPS-SipAddress
    SPS-DisplayOrder
    SPS-Dotted-line
    Office
    UserName                        ingo
    SPS-FeedIdentifier              https://intranet13.sharepoint.farm/personal/
↪ingo;1.a2cf9b4cd9714727bf0b18876689e4b7.243d067e81744aad96c6ee849a72c3e4.b8a0...
    Email
    Fax
    SPS-UserPrincipalName           ingo@sharepoint.farm
    SPS-PicturePlaceholderState
    Peers                           {}
    SPS-ResourceSID
    ExtendedManagers                {}
    SPS-PhoneticLastName
    SID                             S-1-5-21-332261926-3817531067-2902869885-
↪1115
    PictureURL
    SPS-DistinguishedName           CN=Ingo,OU=kenaflow,DC=sharepoint,DC=farm
```

14.5.39.5 Parameters

`[-Item <list-item-object>]`

The item to be used. If nothing is specified here the currently processed list item is used. In a *site* workflow there is no list item. Then you have to specify it yourself. (See 3rd example below.)

`-UserField <field-name>`

The name of a user field.

`-UserName <user-name>`

The name of a user. sAMAccountName with domain or user principal name.

`-User <User-object>`

A SharePoint user object.

`-UserFieldValue <FieldUserValue-object>`

A SharePoint user field value (Type `Microsoft.SharePoint.Client.FieldUserValue`).

`-UserId <user-id>`

The SharePoint site specific id of a user.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.39.6 Example

```
Get-KFUserInfo -UserField "Author"
```

```
Get-KFUserInfo -UserFieldValue $item["Author"]
```

```
$item = Get-PnpListItem -List "Custom List" -Id 1
$user = Get-KFUser "{{Author}}" -Item $item
Get-KFUserInfo -UserId $user
```

14.5.39.7 Related Cmdlets

- *Get-KFUsers*

14.5.40 Get-KFUsers

14.5.40.1 Syntax

```
Get-KFUsers
```

14.5.40.2 Description

14.5.41 Import-KFExcelFile

14.5.41.1 Syntax

```
Import-KFExcelFile
  -InFile <excel-file>
  -SheetName <string>
  [-Address <address>]
  [-Fields <list-of-string>]
  [-AfterReadingCell <scriptblock>]
  [-AfterReadingHeaderCell <scriptblock>]
  [-ThrowErrors[:$true]]
```

14.5.41.2 Support

>= *kenaflow* 4.0.11

14.5.41.3 Return

`System.Collections.Hashtable` object or `$null`

14.5.41.4 Description

This cmdlet can be used to import an Microsoft Excel file. It enables the workflow developer to consume data from an Excel file source.

The result is returned as `Hashtable` object. The keys are taken from the first line of the imported Excel worksheet `-SheetName`, starting with the given `-Address` or cell `A1`.

The input Excel file is given as

1. as System.IO.Stream
2. as Base64 encoded byte array
3. or as file on disk, relative to the workflow folder.

With `-Fields` the imported fields can be filtered. If this parameter is present it's values must match the string values in the cells of the first line of the imported worksheet starting at position `A1` or starting at the specified position in `-Address`.

There are two script blocks that can be used to post-process the imported data before writing it to the result hashtable object:

1. `-AfterReadingHeaderCell` is executed after reading every header cell in the first line of the imported worksheet. It's is used to modify the hashtable keys. There is an example below.
2. `-AfterReadingCell` is executed after reading every data cell from the imported worksheet. It can be used to modify data before it is written to the result hashtable. There is an example below.

14.5.41.5 Parameters

`-InFile <template-excel-file>`

An Excel file to be imported.

1. The file can be specified as Base64 encoded string or
2. as System.IO.Sream object or
3. as a full or relative path to an existing file on disk. It's relative to the current workflow folder.

`-SheetName <string>`

The name of the Excel worksheet.

`[-Address <address>]`

This parameter can be used to specify the worksheet region for the data import.

`<address>` can be

- an Excel style cell addres, such as `A1` or `ZZ55`.
- an Hashtable of this kind: `@{Row="4"; Col="2"}` (The Excel address of the cell would be `B4`.)

`[-Fields <list-of-strings>]`

With `-Fields` the imported fields can be filtered. If this parameter is present it's values must match the string values in the cells of the first line of the imported worksheet starting at position `A1` or starting at the specified position in `-Address`.

`[-AfterReadingHeaderCell <scriptblock>]`

This scriptblock is executed after reading every header cell in the first line of the imported worksheet. It's is used to modify the hashtable keys.

This is an example of who the script show look like, especially to get the corret parameters. The script MUST return a value!

```
-AfterReadingHeaderCell {
    param($cell, $row, $col, $currentFieldName)
    return $currentFieldName
}
```

```
>

`[-AfterReadingCell <scriptblock>]`
> This scriptblock is executed after reading every data cell from the imported␣
↪worksheet. It can be used to modify data before it is written to the result␣
↪hashtable.
>
> This is an example of who the script show look like, especially to get the␣
↪corret parameters. The script MUST return a value!
> ```powershell
    -AfterReadingCell {
        param($cell, $row, $col, $currentValue)
        return $currentValue
    }
```

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.41.6 Example

14.5.41.6.1 1.

This example show how to read an Excel file from disk. It additionaly show how to manipulate the imported data.

```
    $ht2 = Import-KFExcelFile -InFile "$PSScriptRoot\inp3.xlsx" -SheetName
↪"Tabelle1" -Address "A1" -AfterReadingHeaderCell {
        param($cell, $row, $col, $currentFieldName)

        write-host $currentFieldName

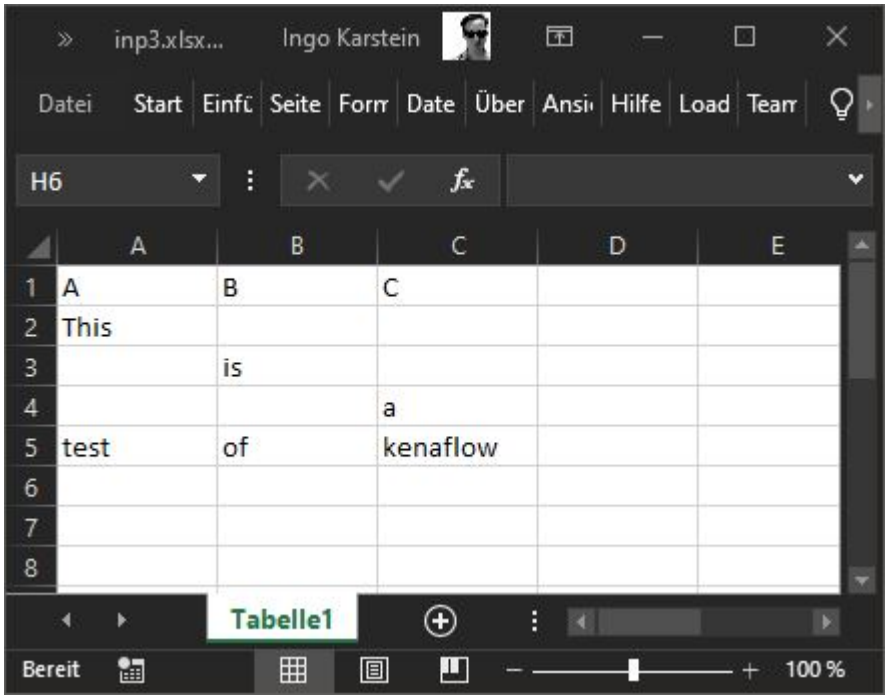
        return "Field $currentFieldName"
    } -AfterReadingCell {
        param($cell, $row, $col, $currentValue)

        write-host $currentValue

        return "x $currentValue"
    }

    $ht2 | % {
        write-host "----Hashtable start"
        $_
    }
```

Input file:



Output:

```
    ----Hashtable start
    Field A                         x This
    ----Hashtable start
    Field B                         x is
    ----Hashtable start
    Field C                         x a
    ----Hashtable start
    Field C                         x kenaflow
    Field B                         x of
    Field A                         x test
```

14.5.41.6.2 2.

This script combines `Import-KFExcelFile` with *`New-KFExcelFile`*. The Excel file is created in memory. The `DateTime` values in Excel can be converted to regular System.DateTime values using `[DateTime]::FromOADate`.

```
    $i = New-KFExcelFile -input @(
            @{FieldA = "kenaflow"; FieldB = [DateTime]::Now; FieldC = 10.101}
            @{FieldA = "is"; FieldB = [DateTime]::Now.AddDays(1); FieldC = 10.101}
            @{FieldA = "cool"; FieldB = [DateTime]::Now.AddMonths(1); FieldC = 10.
↪101}
            @{FieldA = "!"; FieldB = [DateTime]::Now.AddYears(1); FieldC = 10.101}
    ) -asbase64  -SheetName "test1"

    $ht = Import-KFExcelFile -InFile $i -SheetName "test1" -Address @{Col=1;Row=1}

    $ht = Import-KFExcelFile -InFile $i -SheetName "test1" -Address "A1"

    $ht
```

Output:

```
Name                           Value
----                           -----
FieldB                         44193,4891746759
FieldC                         10,101
FieldA                         kenaflow
FieldB                         44194,4891746759
FieldC                         10,101
FieldA                         is
FieldB                         44224,4891746759
FieldC                         10,101
FieldA                         cool
FieldB                         44558,4891746759
FieldC                         10,101
FieldA                         !
```

14.5.41.7 Related Cmdlets

- *Invoke-KFTemplate*
- *Invoke-KFFileTemplate*
- *New-KFPDF*
- *Import-KFExcelFile*
- *Update-KFExcelFile*

14.5.42 Invoke-Kenaflow

14.5.42.1 Syntax

```
Invoke-Kenaflow
  [-WFConfig <path-to-wfconfig>]
  [-EventData <scriptblock>]
  [-ItemId <list-item-id-as-int>]
  [-List <list-title>]
  [-Web <web-url>]
  [-AllStates[:$false]]
  [-DumpConfigValues[:$false]]
  [-Breakpoint[:$false]]
  [-NoPreQueryScript[:$false]]
  [-IgnoreTBE[:$true]]
  [-IgnoreDebugBreaks[:$false]]
  [-ErrorActionPreference <kenaflow-error-action-preference> (Stop)]
  [-ThrowErrors[:$true]]
```

14.5.42.2 Support

>= *kenaflow* 1.0.0

14.5.42.3 Return

Nothing.

BUT...

- In case of "unstructured data" remote event debugging it could return data - if the workflows is configured to do so.
- In case of "link events" that request an answer from the workflow the return object is a HTML string.

14.5.42.4 Description

This cmdlet is used to initialize the *kenaflow* runtime for debugging.

Therefore it's used only in PowerShell development applications such as "PowerShell ISE" or "Visual Studio Code".

If part of the "Magic Line":

```
    if($wf-eq$null){Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";
↪Invoke-Kenaflow;exit}
```

We have described debugging in this article: *Debugging*. Please read it for details.

Use the `-PreWorkflowExecution` parameter to specify a script block to execute before running the workflow script to debug. For example you can initialize variables with this parameter. It is described in article *Debugging*.

Use the `-WFConfig` parameter to specify the path to the workflow configuration file `_wfconfig.ps1`, which should be used to initialize *kenaflow* for debugging the workflow script file. If the parameter is not specified, the workflow configuration file is read from the same directory in which the workflow script to be debugged is located.

The parameter `-EventData` can be used to configure the debug environment to execute remote events. They are "simulated" as described in article *Debugging*.

With the `-ItemId` parameter, *kenaflow* can be forced during debugging to load a specific list item and pass it to the workflow script to be debugged. This may be useful, for example, when the list item is in a certain state or is brought to test a particular workflow situation.

This belongs only to SharePoint *state machine* workflows: With the parameter `-AllStates` you can instruct *kenaflow* to execute the scripts for all workflow states. You could debug them all at once by setting breakpoints in all state scripts. - If **not** specified the only the state script that was started will be executed and the engine (in this debug environment) will skip the other state scripts. The state that the script belongs to is figured out by the workflow configuration.

Use the `-Breakpoint` parameter to tell *kenaflow* to put a PowerShell breakpoint on the first line of the script so that the PowerShell debugger stops there. This can be useful after restarting the IDE when all breakpoints are cleared to prevent *kenaflow* from going through the workflow script completely without stopping. - Nowadays, we recommend to use *Invoke-KFDebugBreak*.

Most workflows have a setting "Time Between Execution" which specifies how much time must elapse between two executions of the workflow. - In debugging this is ignored. But if you want to test it use `-IgnoreTBE:$false`.

14.5.42.5 Parameters

Most of the parameters are described in the "Description" above!

`[-WFConfig <path-to-wfconfig>]`

Described above.

`[-EventData <scriptblock>]`

Described above.

`[-ItemId <list-item-id-as-int>]`

Described above.

`[-List <list-title>]`

In case of SharePoint workflows with multiple lists per web you can specify a filter here for the list title. - The name of the list is case sensitive and must match.

`[-Web <web-url>]`

In case of SharePoint workflows with multiple webs you can specify a filter here for the web URL. The web URL is case insensitive.

`[-AllStates[:$false]]`

Described above.

`[-DumpConfigValues[:$false]]`

If set the engine will write the config values the the *kenaflow* log that is shown in the PowerShell console/output window.

`[-Breakpoint[:$false]]`

Described above.

`[-NoPreQueryScript[:$false]]`

Can be used to disable the execution of the workflow's pre-query script.

`[-IgnoreTBE[:$true]]`

Described above.

`[-IgnoreDebugBreaks[:$false]]`

Described above.

`[-ErrorActionPreference <kenaflow-error-action-preference> (Stop)]`

Described above.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.42.6 Example

```
Invoke-Kenaflow # :-)
```

14.5.43 Invoke-KFCtrlC

14.5.43.1 Syntax

```
Invoke-KFCtrlC [-ThrowErrors[:$true]]
```

14.5.43.2 Support

>= *kenaflow* 1.0.0

14.5.43.3 Return

Nothing.

14.5.43.4 Description

!! This cmdlet is for testing only. It has use in production.

This cmdlet simulates a `Ctrl+C` as it would come from the user if he/she stops the engine with this key stroke.

14.5.43.5 Parameters

`[-ThrowErrors[:$true]]`

> This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.43.6 Example

```
Invoke-KFCtrlC
```

14.5.43.7 Related Cmdlets

- *Invoke-KFGlobalStop*
- *Invoke-KFPipelineStop*
- *Invoke-KFTerminatePosh*

14.5.44 Invoke-KFCustomDataEventSimulation

14.5.44.1 Syntax

```
Invoke-KFCustomDataEventSimulation
  -Json <string>
  [-SimulatedUserName <username> (the current user)]
  [-SimulatedUserPassword <password> ("123")]
  [-MarkAsAuthenticated[:$true]]
  [-Authentication <authentication-method> (Anonymous)]
  [-ThrowErrors[:$true]]
```

14.5.44.2 Support

>= *kenaflow* 2.0.46

14.5.44.3 Return

An object of type `kenaflow.EventDataSimResult`:

```
public class EventDataSimResult
{
        public RemoteEventSource EventSource { get; set; }

        public object EventData { get; set; }

        public int OutputFormat { get; set; } = 0;
}
```

`RemoteEventSource` will have the value `Unstructured` for this cmdlet.

14.5.44.4 Description

This cmdlet is used to simulate a Remote Event to be able to debug workflow scripts in the event execution context.

Please read *Debugging* and *Remote Event* for details!

The cmdlet can only be used in a script block that is specified for parameter `-EventData` of cmdlet *Invoke-Kenaflow*.

The return object of this cmdlet is used by `Invoke-Kenaflow` to configure the runtime environment for *kenaflow* to execute the remote event.

"Custom Data" remote events are used to pass in JSON formatted data in the query string part of the URL with http "GET" method.

The object in the JSON input string MUST have the workflow id as property `__wfid` and the item id the event belongs to as property `__itemid`. BOTH properties are required! The rest of the object is "custom".

14.5.44.5 Parameters

`-Json <string>`

A Json formatted object.

`[-SimulatedUserName <username> (the current user)]`

This is used to simulate an authenticated request. It is passed to the script as part of the `$eventData` object. It is only used in case of 'Basic' authentication. See parameter `-Authentication` below. Default is the name of the current user.

`[-SimulatedUserPassword <password> ("123")]`

This is the simulated password for the authentication. Default is "123".

`[-MarkAsAuthenticated[:$true]]`

This can be used to simulate a not successful authenticated request. It is passed to the script as part of the `$eventData` object.

`[-Authentication <authentication-method> (Anonymous)]`

One of these values:

- `Anonymous` (default)
- `NTML`
- `Basic`

In case of NTLM the current users credentials are used to simulate the request.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.44.6 Example

```
    Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

    Invoke-Kenaflow -Debug -EventData {
        Invoke-KFCustomDataEventSimulation `
                -Json '{"__wfid":"89525426-958f-4630-85bc-06cea78bafe3","__itemid
↪":1,"customData":{"key":"value"}}' `
                    -SimulatedUserName "sharepoint\ingo" -SimulatedUserPassword
↪"kr@ftwerk" -Authentication Basic
    }
```

14.5.44.7 Related Cmdlets

- *Invoke-Kenaflow*
- *Invoke-KFLinkEventSimulation*
- *Invoke-KFSharePointEventSimulation*

14.5.45 Invoke-KFDebugBreak

14.5.45.1 Syntax

```
Invoke-KFDebugBreak
  [-ThrowErrors[:$true]]
```

14.5.45.2 Support

>= *kenaflow* 2.0.0

14.5.45.3 Return

Nothing.

14.5.45.4 Description

This cmdlet is used as "hard coded breakpoint". It does not affect automatic execution of workflows. It does only work during debug. You can leave it in after development, if you want.

14.5.45.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.45.6 Example

```
Invoke-KFDebugBreak
```

14.5.46 Invoke-KFException

14.5.46.1 Syntax

```
Invoke-KFException
   [[-Message] <string>]
   [-SystemException[:$false]]
   [-ThrowErrors[:$true]]
```

14.5.46.2 Support

>= *kenaflow* 4.0.23

14.5.46.3 Return

Nothing.

14.5.46.4 Description

!! This cmdlet is for testing only.

Can be used to throw an exception from within a *kenaflow* cmdlet. The *kenaflow* cmdlet common parameter `-ThrowErrors` is used and therefore the current setting of *`Set-KFThrowErrorsPreference`*.

14.5.46.5 Parameters

`[[-Message] <string>`

Can be used to specify the exception message.

`[-SystemException[:$false]]`

If specified or `:$true` an `System.Exception` will be thrown.

If not specified or `:$false` a *kenaflow* exception of type `kenaflow.KenaflowException` will be thrown.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.46.6 Example

```
Invoke-KFCtrlC
```

14.5.46.7 Related Cmdlets

14.5.47 Invoke-KFFileTemplate

14.5.47.1 Syntax

```
Invoke-KFFileTemplate
  -TemplateStream <System.IO.Stream-object>
  [-TemplateType] <template-type>
  [[-LocalConfig] <local-config>
  [-ToFile <path>]
  [-ToUrl <url>]
  [-ToBytes[$:false]]
  [-ToStream[$:false]]
  [-OverwriteIfExists[:$true]]
  [-ThrowResolveErrors[:$false]]
  [-ThrowErrors[:$true]]
```

```
Invoke-KFFileTemplate
  -TemplateBytes <bytes-array>
  [-TemplateType] <template-type>
  [[-LocalConfig] <local-config>
  [-ToFile <path>]
  [-ToUrl <url>]
  [-ToBytes[$:false]]
```

(continues on next page)

(continued from previous page)

```
    [-ToStream[$:false]]
    [-OverwriteIfExists[:$true]]
    [-ThrowResolveErrors[:$false]]
    [-ThrowErrors[:$true]]
```

```
  Invoke-KFFileTemplate
    -TemplateUrl <url>
    [-TemplateType] <template-type>
    [[-LocalConfig] <local-config>
    [-ToFile <path>]
    [-ToUrl <url>]
    [-ToBytes[$:false]]
    [-ToStream[$:false]]
    [-OverwriteIfExists[:$true]]
    [-ThrowResolveErrors[:$false]]
    [-ThrowErrors[:$true]]
```

```
  Invoke-KFFileTemplate
    -TemplatePath <file-path>
    [-TemplateType] <template-type>
    [[-LocalConfig] <local-config>
    [-ToFile <path>]
    [-ToUrl <url>]
    [-ToBytes[$:false]]
    [-ToStream[$:false]]
    [-OverwriteIfExists[:$true]]
    [-ThrowResolveErrors[:$false]]
    [-ThrowErrors[:$true]]
```

14.5.47.2 Support

>= kenaflow 1.0.0

14.5.47.3 Return

String or Stream or Byte array or nothing.

14.5.47.4 Description

This Cmdlet can be used to process Microsoft Word or PowerPoint files that contain placeholders. They are replaced with content which comes from the workflow *Config List* or the current or per item specified list item, or from a local configuration object that can optionally be transferred with `-LocalConfig`.

Please read the article "*Templates*" for details.

When using `-ThrowResolveErrors:$false` you can suppress exceptions resulting from not existing placeholders.

The file template can

- be a byte array
- be a local file, e.g. in the workflow folder
- be a System.IO.Stream
- be a SharePoint located file

The result file can

- be written as local file, e.g. in the workflow folder
- be written to a SharePoint library (in case of SharePoint workflows)
- be returned as Byte array
- be returned as System.IO.Stream

If multiple return parameters are specify only one is used, in this order (the first that comes is served):

- `-ToFile`
- `-ToStream`
- `-ToBytes`
- `-ToUrl`

14.5.47.5 Parameters

`-TemplateStream <System.IO.Stream-object>`

The input template is a stream.

`-TemplateUrl <url>`

The input template is a SharePoint located file. Only available for SharePoint workflows.

`-TemplatePath <file-path>`

The input template is a file in the file system, e.g. in the workflow folder.

`-TemplateBytes <bytes-array>`

The input template as byte array.

`[-TemplateType] <template-type>`

The type of the template file

- `Word` or
- `PowerPoint`

` [[-LocalConfig]]

The "local config" object that can be used to pass in data as Hashtable to replace placesholders of the format `((xxx))`. This is optional.

` [-Item]

In case of SharePoint workflow you can specify a list item here. In case of *list* and *state machine* workflow the currently processed item is used by default.

`[-ToFile <path>]`

Writes the result to a local file.

`[-ToUrl <url>]`

Writes the result to a SharePoint library. Only available for SharePoint workflows.

`[-ToBytes[$:false]]`

Returns the result as byte array.

`[-ToStream[$:false]]`

Returns the result as stream.

`[-OverwriteIfExists[:$true]]`

Can be used **not** to overwrite the result by specifying `-OverwriteIfExists:$false`

`[-ThrowResolveErrors[:$false]]`

Templates normally contain placeholders. Sometimes they are missing or there is an error while replacing the placeholder. In this case and without this parameter an exception is thrown and the script maybe stops. If you specify this parameter the missing or wrong interpreted placeholders are ignored. !This could lead to format or content errors!

`[-ThrowErrors[:$true]]`

14.5.47.6 Examples

```
    $item = Get-PnpListItem -List "Custom List" -Id 1
    Invoke-KFTemplate "<html><body><span>{{Author}}</span></body></html>" -Item
↪$item -ThrowResolveErrors:$false
```

14.5.47.7 Related Cmdlets

- *Invoke-KFTemplate*
- *New-KFExcelFile*
- *New-KFPDF*
- *Import-KFExcelFile*
- *Update-KFExcelFile*

14.5.48 Invoke-KFGlobalStop

14.5.48.1 Syntax

```
    Invoke-GlobalStop
      [-ThrowErrors[:$true]]
```

14.5.48.2 Support

>= *kenaflow* 2.0.0

14.5.48.3 Return

Nothing.

14.5.48.4 Description

!! This cmdlet is for testing only. It has use in production.

This cmdlet simulates a "global stop" of *kenaflow* as it would come from various situations, e.g. in case of a pending *kenaflow* update that is recognized by the running `kenaflow.exe` process. It sends a process internal "GlobalStop" event to inform all threads to quit their job.

14.5.48.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.48.6 Example

```
Invoke-GlobalStop
```

14.5.48.7 Related Cmdlets

- *Invoke-KFCtrlC*
- *Invoke-KFPipelineStop*
- *Invoke-KFTerminatePosh*

14.5.49 Invoke-KFLinkEventSimulation

14.5.49.1 Syntax

```
Invoke-KFLinkEventSimulation
  -ItemId <id>
  -Data <hashtable>
  [-Validation <hashtable>]
  [-ValidUntil <datetime>]
  [-NTLM[:$false]]
  [-SimulatedUserName <username> (the current user)]
  [-SimulatedUserPassword <password> ("123")]
  [-ThrowErrors[:$true]]
```

14.5.49.2 Support

>= *kenaflow* 2.0.46

14.5.49.3 Return

An object of type `kenaflow.EventDataSimResult`:

```
public class EventDataSimResult
{
        public RemoteEventSource EventSource { get; set; }

        public object EventData { get; set; }

        public int OutputFormat { get; set; } = 0;
}
```

`RemoteEventSource` will have the value `Link` for this cmdlet.

14.5.49.4 Description

This cmdlet is used to simulate a *link* remote event to be able to debug workflow scripts in the event execution context.

Please read *Debugging* and *Remote Event* for details!

The cmdlet can only be used in a script block that is specified for parameter `-EventData` of cmdlet *Invoke-Kenaflow*.

The return object of this cmdlet is used by `Invoke-Kenaflow` to configure the runtime environment for *kenaflow* to execute the remote event.

14.5.49.5 Parameters

`-ItemId <id>`

The id of the SharePoint item the simulared link remote event belongs to.

`-Data <hashtable>`

The field values to be set if the link remote event is considered to be valid.

`[-Validation <hashtable>]`

The given object contains field values that MUST ALL match the specified item's current field values to consider the remote event to be valid.

`[-ValidUntil <datetime>]`

If specified the link remote event is only valid until *this* point in time.

`[-MarkAsAuthenticated[:$true]]`

This can be used to simulate a not successful authenticated request. It is passed to the script as part of the `$eventData` object.

`[-NTML[:$false]]`

Currently link events can be anonymous (not authenticated) or authenticated with NTLM. We're working on it to support basic authentication.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

Currently not used (we are working on "Basic Authentication" for link remote events): `[-SimulatedUserName <username> (the current user)]`

This is used to simulate an authenticated request. It is passed to the script as part of the `$eventData` object. It is only used in case of 'Basic' authentication.

`[-SimulatedUserPassword <password> ("123")]`

This is the simulated password for the authentication. Default is "123".

14.5.49.6 Example

```
Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

Invoke-Kenaflow -EventData {
                Invoke-KFLinkEventSimulation `
                        -ItemId 1 -Data @{Title="kenaflow is cool!"} `
                        -Validation @{AControlField="7f53e8e3"} `
                        -SimulatedUserName "sharepoint\ingo" `
                        -SimulatedPassword "kr@ftwerk" `
                        -Data @{Title = "kenaflow is cool!"}
}
```

14.5.49.7 Related Cmdlets

- *Invoke-Kenaflow*
- *Invoke-KFUnstructuredDataEventSimulation*
- *Invoke-KFCustomDataEventSimulation*
- *Invoke-KFSharePointEventSimulation*

14.5.50 Invoke-KFPipelineStop

14.5.50.1 Syntax

```
Invoke-KFPipelineStop
  [-ThrowErrors[:$true]]
```

14.5.50.2 Support

>= *kenaflow* 2.0.0

14.5.50.3 Return

Nothing.

14.5.50.4 Description

!! This cmdlet is for testing only. It has use in production.

This cmdlet simulates a PowerShell “pipeline stop”.

14.5.50.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.50.6 Example

```
Invoke-KFPipelineStop
```

14.5.50.7 Related Cmdlets

- *Invoke-KFCtrlC*
- *Invoke-KFGlobalStop*
- *Invoke-KFTerminatePosh*

14.5.51 Invoke-KFSharePointEventSimulation

14.5.51.1 Syntax

```
Invoke-KFSharePointEventSimulation
  -ItemId <id>
  [-ListTitle <list-title>]
  [-Web <web-url>]
  -SharePointEventType <event-type>
  -Data <hashtable>
  [-LCID <language-code> (1031)]
  [-SimulatedUserName <username> (the current user)]
  [-ThrowErrors[:$true]]
```

14.5.51.2 Support

>= *kenaflow* 2.0.46

14.5.51.3 Return

An object of type `kenaflow.EventDataSimResult`:

```
public class EventDataSimResult
{
        public RemoteEventSource EventSource { get; set; }

        public object EventData { get; set; }

        public int OutputFormat { get; set; } = 0;
}
```

`RemoteEventSource` will have the value `RER` for this cmdlet.

14.5.51.4 Description

This cmdlet is used to simulate a SharePoint Remote Event to be able to debug workflow scripts in the event execution context.

Please read *Debugging* and *Remote Event* for details!

If the workflow has configured multiple webs and/or lists you can filter the web and/or list using `[-ListTitle <list-title>]` and/or `[-Web <web-url>]`. With or without the list and/or web filter at least **the first list from the first web** will be used as taken from `_wfconfig.ps1`. So please make sure you use the filter to test the event on the right list of the right web!

The cmdlet can only be used in a script block that is specified for parameter `-EventData` of cmdlet *Invoke-Kenaflow*.

The return object of this cmdlet is used by `Invoke-Kenaflow` to configure the runtime environment for *kenaflow* to execute the remote event.

14.5.51.5 Parameters

`-ItemId <id>`

The id of the SharePoint item the simulared remote event belongs to.

`[-ListTitle <list-title>]`

In case multiple lists you can filter the list by its title by using this parameter.

`[-Web <web-url>]`

In case multiple webs you can filter the web by its URL by using this parameter.

`-SharePointEventType <event-type>`

One of the values `ItemAdded` and `ItemUpdated`. It indicates the reason of the SharePoint remote event.

`-Data <hashtable>`

A SharePoint remove event (XML) structure contains the changed items fields. Here you specify the item field values to be passed to the workflow script as part of the `$eventData` object. Remember: we simulate a SharePoint remote event here. It's not real!

`[-LCID <language-code> (1031)]`

This is the language code that is part of the SharePoint remote event structure

`[-SimulatedUserName <username> (the current user)]`

This is used to simulate an authenticated request. It is passed to the script as part of the `$eventData` object. It is only used in case of 'Basic' authentication. See parameter `-Authentication` below. Default is the name of the current user.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.51.6 Example

```
    Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

    Invoke-Kenaflow -EventData {
                    Invoke-KFSharePointEventSimulation `
                            -ItemId 1 `
                            -EventType ItemUpdated `
                            -SimulatedUserName "sharepoint\ingo" `
                            -Data @{Title = "kenaflow is cool!"}
    }
```

14.5.51.7 Related Cmdlets

- *Invoke-Kenaflow*
- *Invoke-KFUnstructuredDataEventSimulation*
- *Invoke-KFLinkEventSimulation*
- *Invoke-KFCustomDataEventSimulation*
- *Invoke-KFSharePointEventSimulation*

14.5.52 Invoke-KFTemplate

14.5.52.1 Syntax

```
Invoke-KFTemplate
  [-Template] <string>
  [[-LocalConfig] <powershell-hashtable-object>]
  [-Item <sharepoint-list-item-object>]
  [-ThrowResolveErrors[:$false]]
  [-ThrowErrors[:$true]]
```

14.5.52.2 Support

>= kenaflow 1.0.0

14.5.52.3 Return

String.

14.5.52.4 Description

This Cmdlet can be used to create template strings with concrete contents which comes from the workflow *Config List* or the current or per item specified list item, or from a local configuration object that can optionally be transferred with `-LocalConfig`.

Please read the article "*Templates*" for details.

When using `-ThrowResolveErrors:$false` you can suppress exceptions resulting from not existing placeholders.

14.5.52.5 Parameters

` [-Template]

The template string.

` [[-LocalConfig]]

The "local config" object that can be used to pass in data as Hashtable to replace placesholders of the format `((xxx))`. This is optional.

` [-Item]

In case of SharePoint workflow you can specify a list item here. In case of *list* and *state machine* workflow the currently processed item is used by default.

`[-ThrowResolveErrors[:$false]]`

Templates normally contain placeholders. Sometimes they are missing or there is an error while replacing the placeholder. In this case and without this parameter an exception is thrown and the script maybe stops. If you specify this parameter the missing or wrong interpreted placeholders are ignored. !This could lead to format or content errors!

```
[-ThrowErrors[:$true]]
```

14.5.52.6 Examples

```
    $item = Get-PnpListItem -List "Custom List" -Id 1
    Invoke-KFTemplate "<html><body><span>{{Author}}</span></body></html>" -Item
↪$item -ThrowResolveErrors:$false
```

14.5.52.7 Related Cmdlets

- *Invoke-KFFileTemplate*
- *New-KFExcelFile*
- *New-KFPDF*
- *Import-KFExcelFile*
- *Update-KFExcelFile*

14.5.53 Invoke-KFTerminatePosh

14.5.53.1 Syntax

```
    Invoke-KFTerminatePosh [-ThrowErrors[:$true]]
```

14.5.53.2 Support

>= *kenaflow* 1.0.0

14.5.53.3 Return

Nothing.

14.5.53.4 Description

!! This cmdlet is for testing only. It has use in production.

This cmdlet forces the PowerShell runtime to stop. It can be used to simulate a "termination" like "Stop Debug" in a PowerShell debugger.

14.5.53.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.53.6 Example

```
Invoke-KFTerminatePosh
```

14.5.53.7 Related Cmdlets

- *Invoke-KFCtrlC*
- *Invoke-KFGlobalStop*
- *Invoke-KFPipelineStop*

14.5.54 Invoke-KFUnstructuredDataEventSimulation

14.5.54.1 Syntax

```
Invoke-KFUnstructuredDataEventSimulation
  -Body <http-request-body>
  [-HttpMethod <method> (POST)]
  [-QueryString <querystring> ($null)]
  [-SimulatedUserName <username> (the current user)]
  [-SimulatedUserPassword <password> ("123")]
  [-MarkAsAuthenticated[:$true]]
  [-Authentication <authentication-method> (Anonymous)]
  [-OutputFormat <output-format> (None)]
  [-ThrowErrors[:$true]]
```

14.5.54.2 Support

>= *kenaflow* 2.0.46

14.5.54.3 Return

An object of type `kenaflow.EventDataSimResult`:

```
public class EventDataSimResult
{
        public RemoteEventSource EventSource { get; set; }

        public object EventData { get; set; }

        public int OutputFormat { get; set; } = 0;
}
```

`RemoteEventSource` will have the value `Unstructured` for this cmdlet.

14.5.54.4 Description

This cmdlet is used to simulate a Remote Event to be able to debug workflow scripts in the event execution context.

Please read *Debugging* and *Remote Event* for details!

The cmdlet can only be used in a script block that is specified for parameter `-EventData` of cmdlet *Invoke-Kenaflow*.

The return object of this cmdlet is used by `Invoke-Kenaflow` to configure the runtime environment for *kenaflow* to execute the remote event.

14.5.54.5 Parameters

`-Body <http-request-body>`

The http request body as string. The data is passed to the workflow script as part of the `$eventData` object.

`[-QueryString <querystring> ($null)]`

This is the query portion of the URL that is passed to the workflow script as part of the `$eventData` object. If it is well formed (`key1=value1&key2=value2&...`) it is split into key-value-pairs. Otherwise the script gets it as string.

`[-HttpMethod <method> (POST)]`

Here you can specify the http method. GET and POST are allowed. POST is default.

`[-SimulatedUserName <username> (the current user)]`

This is used to simulate an authenticated request. It is passed to the script as part of the `$eventData` object. It is only used in case of 'Basic' authentication. See parameter `-Authentication` below. Default is the name of the current user.

`[-SimulatedUserPassword <password> ("123")]`

This is the simulated password for the authentication. Default is "123".

`[-MarkAsAuthenticated[:$true]]`

This can be used to simulate a not successful authenticated request. It is passed to the script as part of the `$eventData` object.

`[-Authentication <authentication-method> (Anonymous)]`

One of these values:

- `Anonymous` (default)
- `NTML`
- `Basic`

In case of NTLM the current users credentials are used to simulate the request.

`[-OutputFormat <output-format> (None)]`

Unstrucuted data remote events can return data to the client. In the request simulation this return data is send to the calling script as return value of `Invoke-Kenaflow`. You can choose one of the following values.

- `None` (default): The workflow script does not return any data to the client.
- `Unconverted`: The return data is not converted in any way. Its passed to the calling script as it is collected by *kenaflow*. This method is for debugging purpose only. To simulate "real" return data use one of the next two options.
- `JsonString`: The return data is serialized to JSON as it would happen in a real http request.

- `ClixmlString`: The return data is serialized to PowerShell CliXML as it would happen in a real http request.
- `Binary`: The return data is written as pure bytes to the http response stream.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.54.6 Example

```
    Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll";

    $output = Invoke-Kenaflow -Debug -EventData {
        Invoke-KFUnstructuredDataEventSimulation -QueryString "test1&test2" -Body
↪"test" `
                    -SimulatedUserName "sharepoint\ingo" -SimulatedUserPassword
↪"kr@ftwerk" -Authentication Basic `
                    -OutputFormat Unconverted
    }

    $output.Test
    $output.currentDate
    $output.Evt
```

14.5.54.6.1 Example "Get-PDF"

When opening the URL in the browser you will see a live generated PDF.

`_wfconfig.ps1`

```
    @{
        Version = "2.0";
        Enabled = $true;
        Type = "POWERSHELL";
        Name = "get-pdf";
        Id = "7c24e165-21c2-418e-ab3e-c6702837b533";
        TBE = -1;
        ConfigListDefaults = @{
            "kenaflow" = "cool"
        };
        AlertAddresses = @();
        MailFrom = "";
        MailReply = "";
        Script = "script.ps1";
        RER = $true;
    }

    if(!$kenaflow){import-module "D:\kenaflow\engine\kenaflow.runtime.dll";Test-
↪KFConfig;exit}
```

`script.ps1`

```
    param($wf, $eventData, $config)
    if($wf-eq$null){import-module "D:\kenaflow\engine\kenaflow.runtime.dll";Invoke-
↪Kenaflow;exit}

    if($eventData) {
```

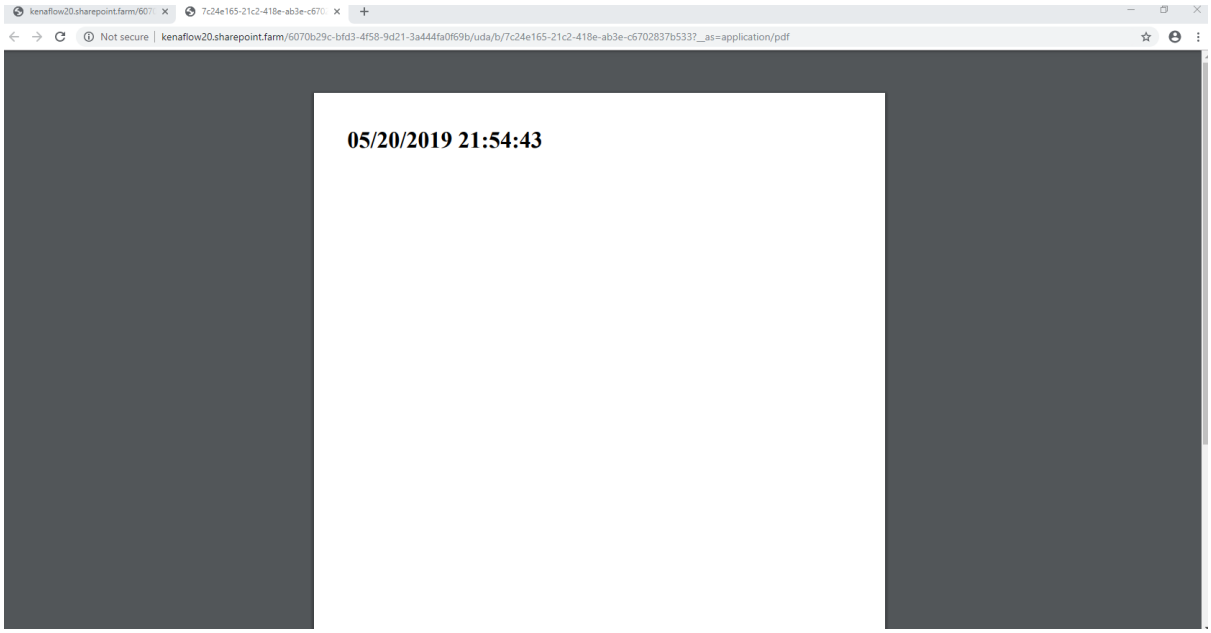
(continues on next page)

(continued from previous page)

```
            return (New-KFPDF -html "<html><body><h1>$([Datetime]::Now)</h1></body>
↪</html>")
    }
```

test.bat

```
    "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" "http://
↪kenaflow20.sharepoint.farm/6070b29c-bfd3-4f58-9d21-3a444fa0f69b/uda/b/7c24e165-
↪21c2-418e-ab3e-c6702837b533?__as=application/pdf"
```



14.5.54.7 Related Cmdlets

- *Invoke-Kenaflow*
- *Invoke-KFLinkEventSimulation*
- *Invoke-KFCustomDataEventSimulation*
- *Invoke-KFSharePointEventSimulation*

14.5.55 Invoke-KFWorkflow

14.5.55.1 Syntax

```
    Invoke-KFWorkflow
      [-WorkflowID] <workflow-id-as-string>
      [-ItemID] <item-id> ($null)
      [-List <list-title>]
      [-Web <web-url>]
      [-Data <object> ($null)]
      [-WithAnswer[:$false]]
      [-AnswerTimeout <milliseconds> (Int.MaxValue)]
      [-ThrowErrors[:$true]]
```

14.5.55.2 Support

>= *kenaflow* 2.0.0

14.5.55.3 Return

Nothing.

14.5.55.4 Description

This cmdlet enables the workflow developer to trigger another workflow.

Please read article *Trigger*.

If `-WithAnswer` is *not* specified this will only trigger the workflow to start. It does not wait for the start.

If `-WithAnswer` is specified this will trigger the workflow and wait for it to finish.

You can pass data to the called workflow. Therefore use `-Data`. Be sure to specify a serializable object!

In case of a call `-WithAnswer` you can specify `-AnswerTimeout` to tell the calling workflow how long it should wait for the called workflows answer. In case of timeout `$null` is returned.

In case you trigger a **SharePoint list workflow** or **SharePoint state machine workflow** you have to use `-ItemId` to tell the called workflow what list item should be processed. If no list item id is specified the called workflow will do nothing.

14.5.55.5 Parameters

`[-WorkflowID] <workflow-id-as-string>`

The id of the workflow that should run by the trigger.

`[-ItemID] <item-id> ($null)`

ID of the SharePoint list item that should be processed by the triggered workflow.

`[-List <list-title>]`

If you trigger a workflow that has multiple list for the selected web (next parameter) configured you **have to** specify the right list. Its the case sensitive title of the list!

`[-Web <web-url>]`

If you trigger a workflow that has multiple webs configured you **have to** specify the right list. Its the case insensitive url of the web!

`[-Data <object> ($null)]`

Data to be transfered to the called workflow.

`[-WithAnswer[:$false]]`

Tells the cmdlet to wait for an answer.

`[-AnswerTimeout <milliseconds> (Int.MaxValue)]`

The timeout in milliseconds.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.55.6 Example

```
Invoke-KFWorkflow \`
  -WorkflowId "8808aecd-f7e5-444c-a625-0d1506c2e99c" \`
  -WithAnswer -AnswerTimeout 10000
```

14.5.56 Lock-KFItem

14.5.56.1 Syntax

```
Lock-KFItem
  -Item <sharepoint-list-item>
  [-MaxWaitInSeconds <seconds> (5)]
  [-ThrowErrors[:$true]]
```

14.5.56.2 Support

>= *kenaflow* 2.0.3

14.5.56.3 Return

`$null` on failure.

A lock handle (`-ne $null`) on success.

14.5.56.4 Description

Used to lock a SharePoint list item manually.

Please read this article for further details: *Locking*

14.5.56.5 Parameters

`[-Item] <sharepoint-list-item>`

SharePoint list item that needs to be locked.

`[-MaxWaitInSeconds <wait-time> (5)]`

Specified (in seconds), how long the workflow waits for the lock if it already registered.

Default: 5 seconds.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.56.6 Example

```
$item = Get-PnPListItem -List "Custom List" -Id 1
$lock = Lock-KFItem -Item $item -MaxWaitInSeconds 10
if( $lock -ne $null ) {
        Test-KFItemLock -Handle $lock
        Unlock
} else {
        Write-KFLog "Cannot get a handle to the SharePoint list item."
}
```

14.5.57 Lock-KFProcess

14.5.57.1 Syntax

```
Lock-KFProcess
  [-Key] <key>
  [-MaxWaitInSeconds <wait-time> (5)]
  [-ThrowErrors[:$true]]
```

14.5.57.2 Support

>= *kenaflow* 2.0.3

14.5.57.3 Return

`$null` on failure.

A lock handle (`-ne $null`) on success.

14.5.57.4 Description

To synchronize multiple *kenaflow* workflows outside a SharePoint / List item context you can use this cmdlet to gather a lock with a custom key.

Please read this article for further details: *Locking*.

14.5.57.5 Parameters

`[-Key] <key>`

The key that is used by multiple workflows to gather the same (!) lock.

`[-MaxWaitInSeconds <wait-time> (5)]`

Specified (in seconds), how long the workflow waits for the lock if it already registered.

Default: 5 seconds.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.57.6 Example

```
$lock = Lock-KFProcess -Key "TestLockKey" -MaxWaitInSeconds 10
if( $lock -ne $null ) {
        Test-KFProcessLock -Handle $lock
        Unlock-KFProcess $lock
} else {
        Write-KFLog "Cannot get a handle for the given key."
}
```

14.5.57.7 Related Cmdlets

- *Test-KFProcessLock*
- *Unlock-KFProcess*

14.5.58 New-KFBarcode

14.5.58.1 Syntax

```
New-KFBarcode
  -AsStream
  -Data <string>
  [-BarcodeFormat <format> ("QR_CODE")]
  [-Height <int> (100)]
  [-Width <int> (100)]
  [-Margin <int> (10))
  [-PureBarcode[:$false]]
  [-GS1Format[:$false]]
  [-ErrorCorrection <error-correction> ($null)]
  [-BackgroundColor <bg-color> ("White")]
  [-ForegroundColor <fg-color> ("Black")]
  [-OutputImageFormat <output-format> ("png")]
  [-ThrowErrors[:$true]]
```

```
New-KFBarcode
  -AsBase64
  -Data <string>
  [-BarcodeFormat <format> ("QR_CODE")]
  [-Height <int> (100)]
  [-Width <int> (100)]
  [-Margin <int> (10))
  [-PureBarcode[:$false]]
  [-GS1Format[:$false]]
  [-ErrorCorrection <error-correction> ($null)]
  [-BackgroundColor <bg-color> ("White")]
  [-ForegroundColor <fg-color> ("Black")]
  [-OutputImageFormat <output-format> ("png")]
  [-ThrowErrors[:$true]]
```

```
New-KFBarcode
  -AsDataUrl
  -Data <string>
  [-BarcodeFormat <format> ("QR_CODE")]
  [-Height <int> (100)]
  [-Width <int> (100)]
  [-Margin <int> (10))
  [-PureBarcode[:$false]]
```

(continues on next page)

(continued from previous page)

```
    [-GS1Format[:$false]]
    [-ErrorCorrection <error-correction> ($null)]
    [-BackgroundColor <bg-color> ("White")]
    [-ForegroundColor <fg-color> ("Black")]
    [-OutputImageFormat <output-format> ("png")]
    [-ThrowErrors[:$true]]
```

```
  New-KFBarcode
    -AsString
    -Data <string>
    [-BarcodeFormat <format> ("QR_CODE")]
    [-Height <int> (100)]
    [-Width <int> (100)]
    [-Margin <int> (10))
    [-PureBarcode[:$false]]
    [-GS1Format[:$false]]
    [-ErrorCorrection <error-correction> ($null)]
    [-BackgroundColor <bg-color> ("White")]
    [-ForegroundColor <fg-color> ("Black")]
    -OutputImageFormat Svg
    [-ThrowErrors[:$true]]
```

14.5.58.2 Support

>= *kenaflow* 2.0.28

14.5.58.3 Return

System.Stream or string.

14.5.58.4 Description

We use the open source library ZXING.NET to generate barcodes from within a *kenaflow* workflow.

The barcode can be returned by the cmdlet as a System.Stream object or a Base64 encoded string that contains the barcode image or as a so called "data URL" that contains the image for easy embedding into HTML.

The data for the barcode is provided with parameter `-data <string>`.

Valid barcode formats are:

- QR_CODE
- AZTEC
- CODABAR
- CODE_39
- CODE_93
- CODE_128
- DATA_MATRIX
- EAN_8
- EAN_13
- ITF
- MAXICODE

- PDF_417
- RSS_14
- RSS_EXPANDED
- UPC_A
- UPC_E
- All_1D
- UPC_EAN_EXTENSION
- MSI
- PLESSEY
- IMB

The default ist "QR_CODE".

`-GS1Format` can be used with "DATA_MATRIX" and "CODE_128".

`-ErrorCorrection <error-correction>` can be used to tell the encode how much redundancy should be included in the barcode image. `<error-correction>` is a string. It's value depends on the barcode type.

For "QR_CODE":

"H" : ~30% correction "Q" : ~25% correction "M" : ~15% correction "L" : ~7% correction

`[-OutputImageFormat <output-format> ("png")]` specifies the image format that is created. Here are the supported types:

- Bmp
- Emf
- Gif
- Icon
- Jpeg
- Png
- Tiff
- Wmf
- Svg (>= _kenaflow _ 4.0.22)

The return value in case of `-AsDataUrl` is a string that contains the MIME type of the image and the image itself as a Base64 encoded string. Here is more information about that: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs There are also some examples.

14.5.58.5 Parameters

`-AsStream`

Specifies the return value type: System.IO.Stream

`-AsBase64`

Specifies the return value type: a string with the Base64 encoded image

`-AsDataUrl`

Specifies the return value type: a data URL with the MIME type and the image itself

`-AsString`

If `OutputImageFormat` is `SVG` this parameter can be used to retrieve the SVG vector image as (unencoded) string

`-Data <string>`

The data that is encoded as barcode

`[-BarcodeFormat <format> ("QR_CODE")]`

The format of the barcode image. Default is "QR_CODE"

`[-Height <int> (100)]`

The height of the barcode image. Default is 100 pixels.

`[-Width <int> (100)]`

The widht of the barcode image. Default is 100 pixels.

`[-Margin <int>)`

The margin (top, left, right, bottom) of the barcode image. If not specified the default value of the barcode creation tool is used. (Before *kenaflow* 4.0.22 it has a default value of `10`.)

`[-PureBarcode[:$false]]`

If set to `$true` only the barcode is contained in the resulting image. No numbers or characters.

`[-GS1Format[:$false]]`

A kind of special format for grouping values in the barcode. Please search the internet for more information.

`[-ErrorCorrection <error-correction> ($null)]`

For "QR_CODE" is specifies the amount of redundancy in the resulting image.

`[-BackgroundColor <bg-color> ("White")]`

The color of the image background as System.Drawing.Color structure. Default is "System.Drawing.Color.White"

`[-ForegroundColor <fg-color> ("Black")]`

The color of the barcode pixels as System.Drawing.Color structure. Default is "System.Drawing.Color.Black"

`[-OutputImageFormat <output-format> ("png")]`

The output image format. Please see the "description" section above for details.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.58.6 Example

```
$barcode = Get-KFBarcode -AsBase64 -Data "Hello kenaflow" -ErrorCorrection "M"
Write-KFLog $barcode
```

14.5.58.7 Related Cmdlets

- *Get-KFBarcode*

14.5.59 New-KFExcelFile

14.5.59.1 Syntax

```
New-KFExcelFile
  -Input <list-of-hashtables>
  -SheetName <string>
  -AsStream
  [-InFile <template-excel-file>]
  [-Fields <list-of-strings>]
  [-ThrowErrors[:$true]]

New-KFExcelFile
  -Input <list-of-hashtables>
  -SheetName <string>
  -AsBase64
  [-InFile <template-excel-file>]
  [-Fields <list-of-strings>]
  [-ThrowErrors[:$true]]

New-KFExcelFile
  -Input <list-of-hashtables>
  -SheetName <string>
  -OutFile <path>
  [-InFile <template-excel-file>]
  [-Fields <list-of-strings>]
  [-ThrowErrors[:$true]]
```

14.5.59.2 Support

>= *kenaflow* 4.0.10

14.5.59.3 Return

In case of using `-AsStream` it returns a `System.IO.MemoryStreaam` object.

In case of using `-AsBase64` it returns a string.

In case of using `-OutFile` it returns nothing.

14.5.59.4 Description

This cmdlet can be used to convert a list of PowerShell hashtables to an MS Excel file.

This is a such a list of hashtables:

```
@(
        @{FieldA = "kenaflow"; FieldB = 1; FieldC = [DateTime]::Now}
        @{FieldA = "is" ; FieldB = 2                                ; FieldD =␣
↪10.101}
        @{FieldA = "cool"; FieldB = 3;
)
```

There are three different output options:

1. as System.IO.MemoryString by using `-AsStream`
2. as Base64 encoded byte array using `-AsBase64`
3. or as file on disk using `-OutFile <path>`

You can only use one output option.

You have to specify the input object with `-Input` and the name of the workbook sheet using `-SheetName`.

If **not** using `-Infile` the cmdlet creates a **new** file as output.

If you specify an existing Excel file with `-InFile <template-excel-file>` you have to 3 options:

1. Specify the input file content as Base64 encoded string or
2. specify the input file as System.IO.Sream object or
3. speficy a full or relative path to an existing file on disk. It's relative to the current workflow folder.

With the input file you can do two things:

1. If the worksheet named with `-SheetName` already exists the cmdlet will take the field names from the 1st row. Than it will take the formatting for the data rows from the 2nd row. Everything starting with row 2 will be removed from the excel file. But no other worksheet is modified! The cell formats are not modified.
2. If the worksheet named with `-SheetName` does not exist it will be added using the default styling.

14.5.59.5 Parameters

`-Input <list-of-hashtables>`

The inpupt object: a List of PowerShell hashtables

`-SheetName <string>`

The name of the Excel worksheet.

`-AsStream`

The result is a System.IO.MemoryStream object

`-AsBase64`

The result is a Base64 encoded byte array that represents the Excel file.

`-OutFile <path>`

The result is written to a file on disk. The path can be relative to the current workflow folder.

`[-InFile <template-excel-file>]`

An Excel file as template. The new worksheet will be added or the existing worksheet will be modified.

1. The template can be specified as Base64 encoded string or
2. as System.IO.Sream object or
3. as a full or relative path to an existing file on disk. It's relative to the current workflow folder.

`[-Fields <list-of-strings>]`

The field order can be specified here. If used only the specified fields are used as keys and read from the hastable objects in the list specified with `-Input`.

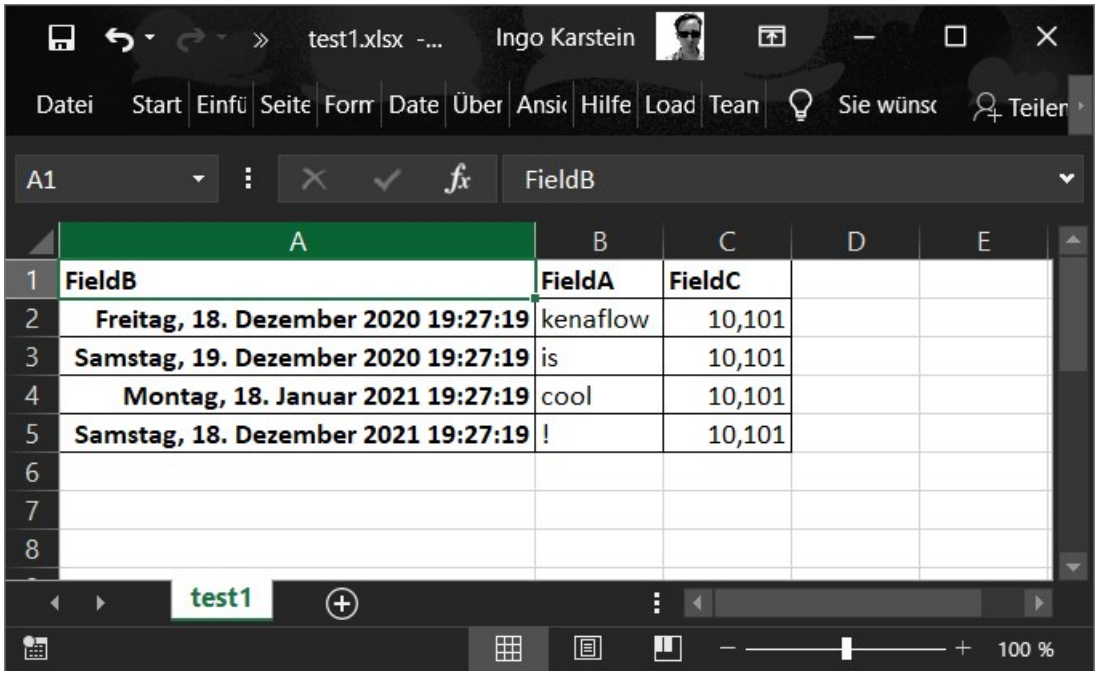
`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.59.6 Example

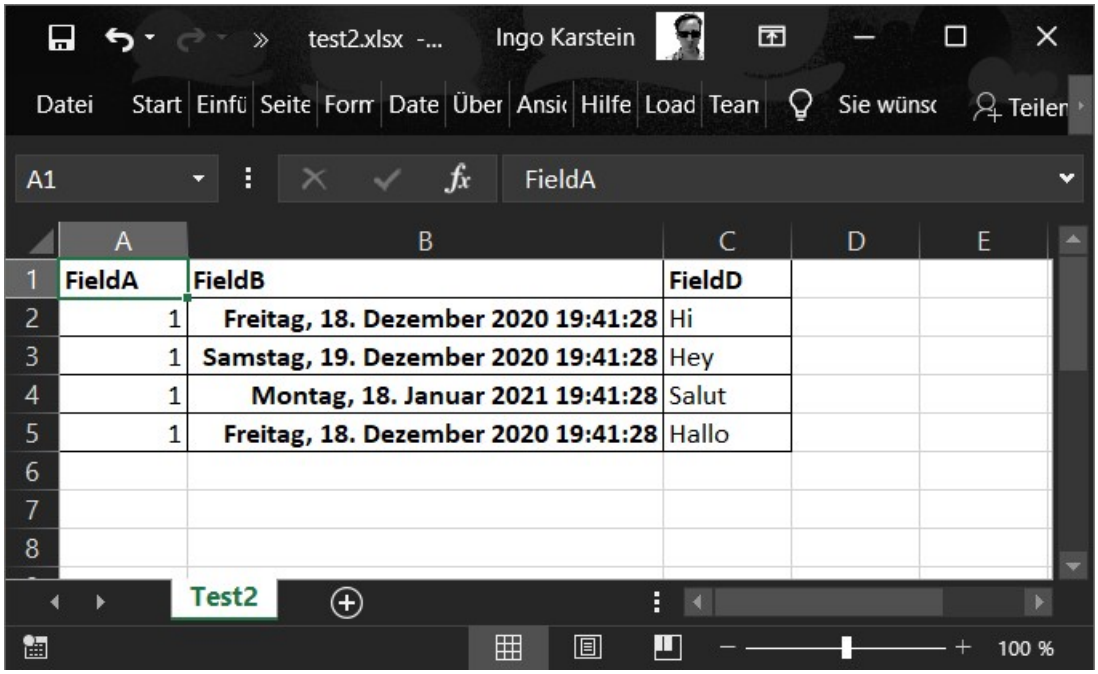
```
    New-KFExcelFile -input @(
            @{FieldA = "kenaflow"; FieldB = [DateTime]::Now; FieldC = 10.101}
            @{FieldA = "is"; FieldB = [DateTime]::Now.AddDays(1); FieldC = 10.101}
            @{FieldA = "cool"; FieldB = [DateTime]::Now.addmonths(1); FieldC = 10.
↪101}
            @{FieldA = "!"; FieldB = [DateTime]::Now.addyears(1); FieldC = 10.101}
    ) -OutFile "test1.xlsx"  -SheetName "test1"
```

Result:



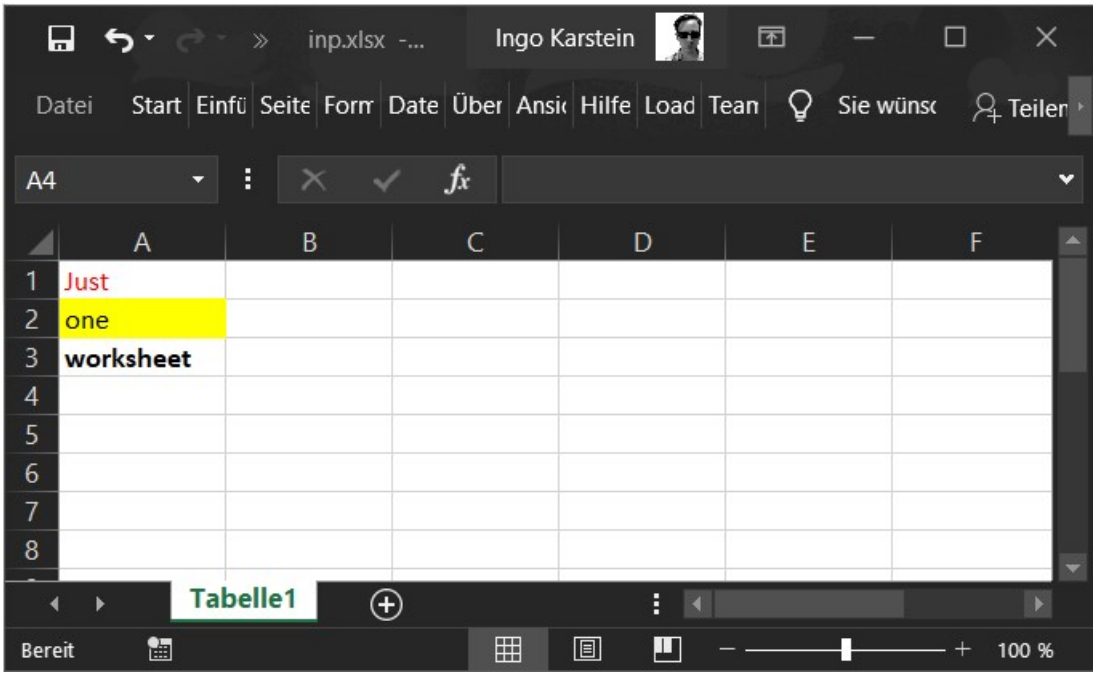

```
    New-KFExcelFile -input @(
            @{FieldA = 1; FieldB = [DateTime]::Now; FieldC = 10.101; FieldD = "Hi"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddDays(1); FieldC = 10.101;␣
↪FieldD = "Hey"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddMonths(1); FieldC = 10.101;␣
↪FieldD = "Salut"}
            @{FieldA = 1; FieldB = [DateTime]::Now; FieldC = 10.101; FieldD =
↪"Hallo"}
    ) -Fields @("FieldA", "FieldB", "FieldD") -OutFile "test2.xlsx" -SheetName
↪"Test2"
```

Result:

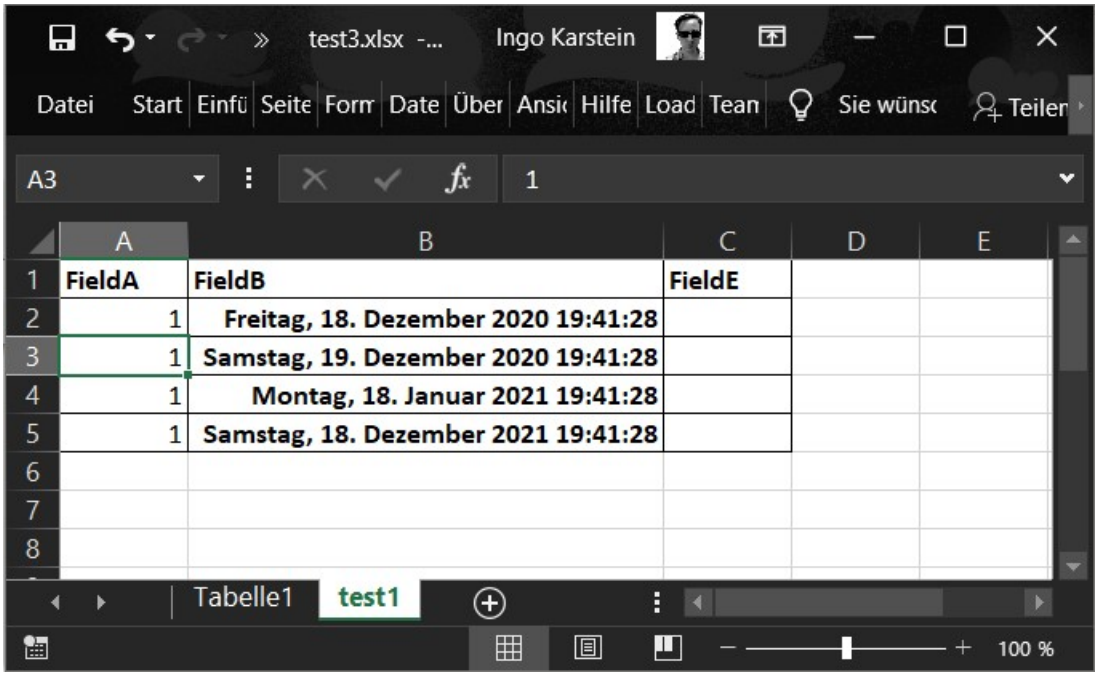



```
    New-KFExcelFile -input @(
            @{FieldA = 1; FieldB = [DateTime]::Now; FieldC = 10.101; FieldD = "Hi"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddDays(1); FieldC = 10.101;␣
↪FieldD = "Hey"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddMonths(1); FieldC = 10.101;␣
↪FieldD = "Salut"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddYears(1); FieldC = 10.101;␣
↪FieldD = "Hallo"}
    ) -Fields @("FieldA", "FieldB", "FieldE") -OutFile "test3.xlsx" -InFile "
↪$PsscriptRoot\inp.xlsx" -SheetName "test1"
```

Template file:

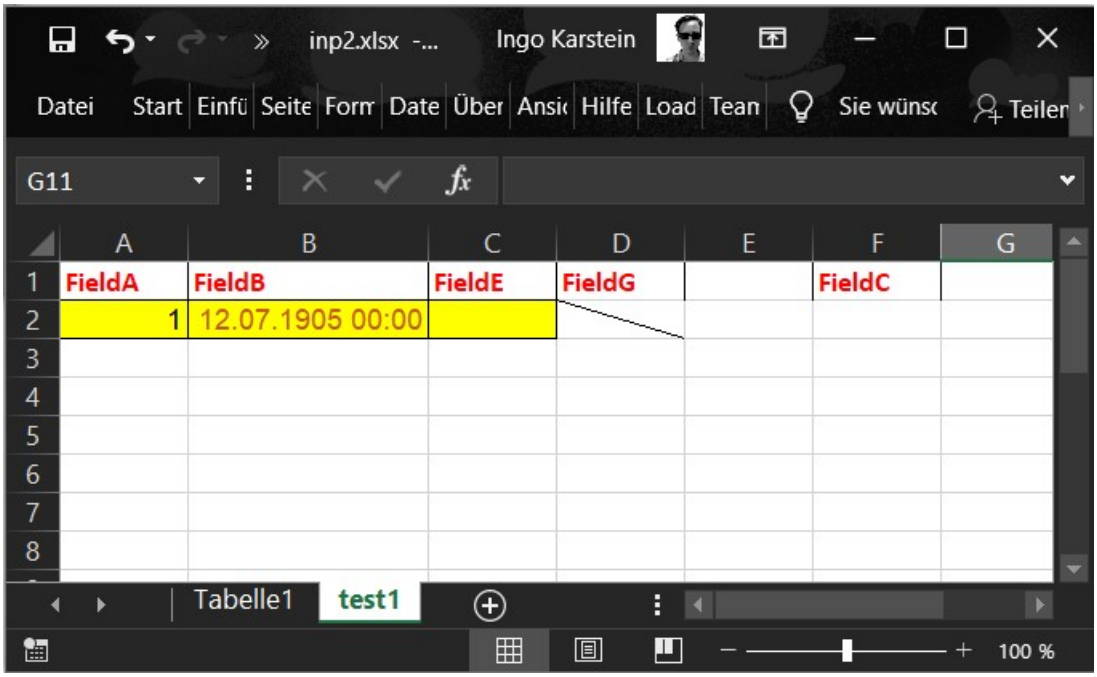


Result:

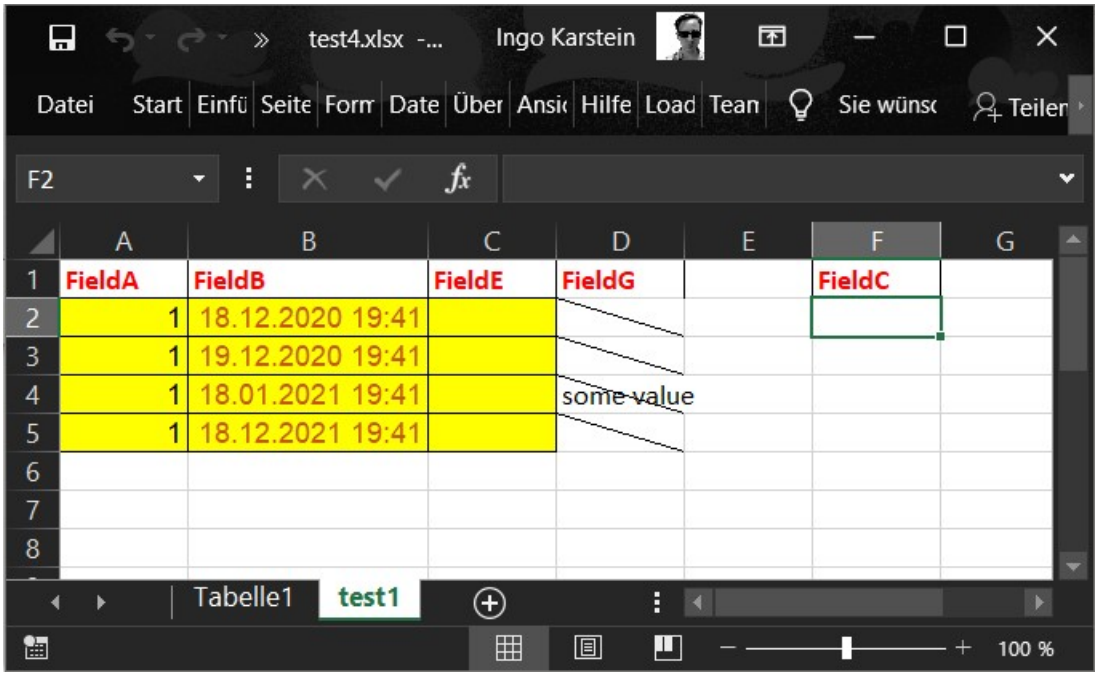



```
    New-KFExcelFile -input @(
            @{FieldA = 1; FieldB = [DateTime]::Now; FieldC = 10.101; FieldD = "Hi"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddDays(1); FieldC = 10.101;␣
↪FieldD = "Hey"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddMonths(1); FieldC = 10.101;␣
↪FieldD = "Salut"; FieldG="some value"}
            @{FieldA = 1; FieldB = [DateTime]::Now.AddYears(1); FieldC = 10.101;␣
↪FieldD = "Hallo"}
    ) -OutFile "test4.xlsx" -InFile "$PsscriptRoot\inp2.xlsx" -SheetName "test1"
```

Template File



Result



14.5.59.7 Related Cmdlets

- *Invoke-KFTemplate*
- *Invoke-KFFileTemplate*
- *New-KFPDF*
- *Import-KFExcelFile*
- *Update-KFExcelFile*

14.5.60 New-KFPDF

14.5.60.1 Syntax

```
New-KFPDF
  -HTML <string>
  [-AsStream[:$false]]
  [-ThrowErrors[:$true]]
```

14.5.60.2 Support

>= *kenaflow* 1.0.0

14.5.60.3 Return

System.IO.Stream or System.Byte array.

14.5.60.4 Description

One of the great advantages of *kenaflow* is the integrated PDF engine, which, together with the template engine, makes it very easy to generate a PDF from simple HTML and the data from the workflow or item.

We use the open source software Chromium Embedded Framework and the .NET port "CEFSharp" to provide this functionality.

The example below shows how to create a PDF document from a template and upload it to SharePoint.

14.5.60.5 Parameters

`-HTML <string>`

The HTML string that gets converted to PDF.

`[-AsStream[:$false]]`

Tells the engine to return a System.IO.Stream object rather than a System.Byte array. Default (if not specified): System.Byte array.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.60.6 Example

```
$Text = (Invoke-KFTemplate -Template "<p>Hello <b>{{Author}}</b>.</p>")
$Pdf = New-KFPDF -HTML $Text -AsStream
Add-PnPFile -Folder "/Archive" -FileName "$($item.ID).pdf" -Stream $Pdf
```

14.5.60.7 Related Cmdlets

- *Invoke-KFTemplate*
- *Invoke-KFFileTemplate*
- *New-KFExcelFile*
- *Import-KFExcelFile*
- *Update-KFExcelFile*

14.5.61 Protect-KFData

14.5.61.1 Syntax

```
Protect-KFData
  -String <string>
  [-ThrowErrors[:$true]]
```

```
Protect-KFData
  -Bytes <bytes-as-array>
  [-ThrowErrors[:$true]]
```

```
Protect-KFData
  -Object <object>
  [-ThrowErrors[:$true]]
```

14.5.61.2 Support

>= *kenaflow* 2.0.19

14.5.61.3 Return

Encrypted data as Base64 encoded string.

(This string start with one of the characters "s" (for "string"), "b" (for "bytes") or "o" (for object).)

14.5.61.4 Description

This cmdlet can be used to encrypt data.

The `EncryptionPassphrase` from the global configuration is used for encryption.

If `-Object` is specified the given `<object>` will be serialized as PowerShell CliXML before encryption.

14.5.61.5 Parameters

`-String <string>`

This `<string>` will be encrypted.

`-Bytes <bytes-as-array>`

The given System.Bytes array will be encrypted.

`-Object <object>`

The specified object will be serialized as CliXML an then encrypted.

14.5.61.6 Examples

```
$encryptedData = Protect-KFData -Object @{Hello="kenaflow"}
Write-KFLogOutput (Unprotect-KFData $encryptedData)
```

14.5.61.7 Related Cmdlets

- *Unprotect-KFData*

14.5.62 Remove-KFData

14.5.62.1 Syntax

```
Remove-KFData
  [-Key] <string>
  [-ThrowErrors[:$true]]
```

14.5.62.2 Support

>= *kenaflow* 1.0.0

14.5.62.3 Return

Nothing.

14.5.62.4 Description

This cmdlet can be used to delete the data previously persisted in the workflow data list with *Set-KFData*.

14.5.62.5 Parameters

`[-Key] <string>`

The key that represents the data that was persisted before.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.62.6 Example

```
    Set-KFData -Key "kenaflow" -Data @{"is" = "a workflow engine for SharePoint,␣
↪Email and PowerShell"}
    Write-KFLog (Get-KFData -Key "kenaflow")
    Remove-KFData -Key "kenaflow"
```

14.5.62.7 Related Cmdlets

- *Get-KFData*
- *Set-KFData*

14.5.63 Remove-KFDBData

14.5.63.1 Syntax

```
    Remove-KFDBData
      [-Key] <key>
      [-RelatedToWorkflow[:$true]]
      [-RelatedToItem[:$true]]
      [-ThrowErrors[:$true]]
```

14.5.63.2 Support

>= *kenaflow* 1.0.0

14.5.63.3 Return

Nothing.

14.5.63.4 Description

This cmdlet removes data perviously with `Set-KFDBData` persisted from the *kenaflow* SQLite database.

We have described this in article *Persisting Data*, section "*kenaflow* SQLite Persistence".

By default `-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` are used - alone or in combination - to automatically add prefix(es) to your key.

14.5.63.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.63.6 Example

```
    Set-KFDBData -Key "kenaflow" -Data @{"is" = "a workflow engine for SharePoint,␣
↪Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:$false
    Write-KFLog (Get-KFDBData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false)
    Remove-KFDBData -Key "kenaflow" -RelatedToItem:$false -RelatedToWorkflow:$false
```

14.5.63.7 Related Cmdlets

- *Set-KFDBData*
- *Get-KFDBData*

14.5.64 Remove-KFItem

14.5.64.1 Syntax

```
Remove-KFItem
  [-ToRecycleBin[:$true]]
  [-ThrowErrors[:$true]]
```

14.5.64.2 Support

>= *kenaflow* 1.0.0

14.5.64.3 Return

Nothing.

14.5.64.4 Description

If this Cmdlet is used in a workflow script, the SharePoint element currently being processed by the workflow will be deleted after the script processing is complete.

The `-ToRecycleBin` parameter can be used to move the item to the Recycle Bin of the Web page collection. This is the default behaviour! If, however, the element is to be permanently deleted, the parameter must be used in this way: `-ToRecycleBin:$false`.

14.5.64.5 Parameters

`[-ToRecycleBin[:$true]]`

Remove item permanently or move it the recycle bin (default).

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.64.6 Example

```
Remove-KFItem -ToRecycleBin:$false
```

14.5.65 Remove-KFItemData

14.5.65.1 Syntax

```
Remove-KFData
  [-Key] <string>
  [-Item <list-item-objekt>]
  [-ThrowErrors[:$true]]
```

14.5.65.2 Support

>= *kenaflow* 1.0.0

14.5.65.3 Return

Nothing.

14.5.65.4 Description

!! This is only available in SharePoint *list* or *state machine* workflows!

This cmdlet can be used to delete data previously persisted with *Set-KFItemData*.

14.5.65.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-Item <list-item-object>]`

The SharePoint list item in whose context the data is serialized.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.65.6 Example

```
    $item = Get-PnPListItem -List "Custom List" -Id 1
    Set-KFItemData -Item $item -Key "kenaflow" -Data @{"is" = "a workflow engine␣
↪for SharePoint, Email and PowerShell"}
    Write-KFLog (Get-KFItemData -Item $item -Key "kenaflow")
    Remove-KFItemData -Item $item -Key "kenaflow"
```

14.5.65.7 Related Cmdlets

- *Get-KFItemData*
- *Set-KFItemData*

14.5.66 Remove-KFObject

14.5.66.1 Syntax

```
    Remove-KFObject
      -Name <string>
      [-ThrowErrors[:$true]]
```

14.5.66.2 Support

>= *kenaflow* 4.0.11

14.5.66.3 Return

Nothing.

14.5.66.4 Description

This cmdlet can be used to remove data from the in-memory-cache previously written by *Set-KFObject*.

14.5.66.5 Parameters

`[-Name] <string>`

The key that is used to write data

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.66.6 Example

The `proProcessingScript` specified in `_wfconfig.ps1` could include the following line of Powershell script code:

```
    Set-KFObject -Name "ListOfDataForExcelExport" -Object @()
```

The SharePoint workflow script could include this script code:

```
    $d = Get-KFObject -Name "ListOfDataForExcelExport"
    $d += @{Title = $item["Title"]; Id = $item.Id; Author = $item["Author"].
↪LookupValue; Editor = $item["Editor"].LookupValue}
    Set-KFObject -Name "ListOfDataForExcelExport" -Object $d
```

This creates an empty list in memory.

The `postProcessingScript` specified in `_wfconfig.ps1` could include the following line of Powershell script code:

```
    $data = Get-KFObject -Name "ListOfDataForExcelExport"
    New-KFExcelFile -Input $data -SheetName "Output" -AsBase64 -Fields @("Id",
↪"Title", "Author", "Editor")

    Remove-KFObject -Name "ListOfDataForExcelExport" # This is not required␣
↪because at the end of a workflow processing the in-memory-object is cleared!
```

14.5.66.7 Related Cmdlets

- *Get-KFObject*
- *Set-KFObject*

14.5.67 Remove-KFPermission

14.5.67.1 Syntax

```
Remove-KFPermission
  [-Permissions] <list-of-permissions>
  [-Item <list-item>]
  [-CopyCurrentPermissions[:$false]]
  [-BreakPermissionInheritance[:$true]]
  [-ThrowResolveErrors[:$false]]
  [-ThrowErrors[:$true]]
```

14.5.67.2 Support

>= *kenaflow* 1.0.0

14.5.67.3 Return

Nothing.

14.5.67.4 Description

Use this cmdlet to remove permissions from a list item.

The `-Permissons` parameter gets a single permission object or a list of permission objects of the following form.

The permission object is specified as a PowerShell Hashtable:

```
@{
   User = <string-or-object>;
   PermissionsSets = @(<permission-set-name>[, <permission-set-name>[, ...]]);
   Quick = "CRUDMA";
)
```

The `User` property of the permission object can be:

- List of strings or objects
- string, that describe the user
- object as (array of) SharePoint `FieldUserValue`

Such a "string" of the `User` property of the permission object can be:

- a user login name
- an email address
- a SharePoint field of the currently processed or with `-Item` specified list item
- a combination of all of them separated by "," or ";"

Examples:

- `User = @($item["Author"])`

- `User = $item["Author"]`
- `User = @($item["Author"], $item["Editor"])`
- `User = "{{Author}}"`
- `User = "{{Author}},{{Editor}}"`
- `User = @($item["Author"], "{{Editor}},sharepoint\ingo", "ingo@sharepoint.farm")`

The *kenaflow* template engine is used to resolve the given strings. *Templates*

Also SharePoint groups are resolved. E.g. `User = @("Intranet Owners")`

The parameter `-PermissionSets` is a PowerShell list of one or more names of valid / existing SharePoint permission sets, such as "Full Control" or "Design". - Custom permission sets can be named too. But the have to exist!

The parameter `-Quick` is a kind of shortcut to `-PermissionSets`. *kenaflow can* create it's own permission sets as specified in the `_wfconfig.ps1` file. They have to be created using program switch *`--createpermissionsets`*. This permission sets are by default named as:

- kenaflow Create => `C`
- kenaflow Read => `R`
- kenaflow Update => `U`
- kenaflow Delete => `D`
- kenaflow Manage => `M`
- kenaflow Approve => `A`

The first letter(s) **in combination** can be used in parameter `-Quick` to specify these permission sets. - But they have to be configured in `_wfconfig.ps1` and must exist in SharePoint! - For example: `Quick="CUD"` => "Create", "Update", "Delete" - but not "Read".

The base permissions of each of the *kenaflow* defined permission sets can be found here: *Permissions*.

This is an example for a valid permission object - if the *kenaflow* permission sets are configured.

```
@{
    User = @("{{Author}}", $item["Editor"]);
    PermissionSets = @("Read");
    Quick = "CUD"
}
```

Multiple permission objects can be specified as a list:

```
@(
    @{User="sharepoint\ingo"; Quick="CUD"},
    @{User="sharepoint\kenaflow";PermissionSets=@("Full Control")}
)
```

14.5.67.5 Parameters

`[-Permissions] <list-of-permissions>`

One or more permission objects as described above.

`[-Item <list-item>]`

The list item to be processed.

`[-BreakPermissionInheritance[:$true]]`

Permissions can only be added to the list item if it has its own permissions - and not inherited permissions from its parent. If an item with active permission inheritance is given the inheritance will be stopped by default. But if you specify `-BreakPermissionInheritance:$false` the cmdlet will fail and stop if the list item has active permission inheritance.

`[-CopyCurrentPermissions[:$false]]`

If the cmdlet stop permission inheritance it can copy the permissions of the parent or start with a blank set of permissions which is the default.

`[-ThrowResolveErrors[:$false]]`

kenaflow tries to resolve the given `User` property of each permission object. If this fails and `-ThrowResolveErrors` is specified the cmdlet will fail. Otherwise the not existing or not successfully resolved principal is skipped.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.67.6 Examples

```
Remove-KFPermission -Permissions @(
    @{
        User = "user1@sharepoint.farm";
        PermissionSets = @("Full Control");
        Quick="MA";
    },
    @{
        User = "user2@sharepoint.farm";
        PermissionSets = "Read";
        Quick="D";
    }
)
```

```
$newEditor = "sharepoint\user3"

Remove-KFPermission -Permissions @(
    @{ User="{{Author}}"; Quick="R" },
    @{ User="{{currentEditor}}"; Quick="R" },
    @{ User=$newEditor; Quick="R" }
)
```

14.5.67.7 Related Cmdlets

- *Add-KFPermission*
- *Get-KFPermission*
- *Reset-KFPermissionInheritance*
- *Set-KFPermission*
- *Stop-KFPermissionInheritance*
- *Update-KFPermission*

14.5.68 Remove-KFSessionStorageData

14.5.68.1 Syntax

```
Remove-KFSessionStorageData
  [-Key] <key>
  [-RelatedToWorkflow[:$true]]
  [-RelatedToItem[:$true]]
  [-ThrowErrors[:$true]]
```

14.5.68.2 Support

>= *kenaflow* 1.0.0

14.5.68.3 Return

Nothing.

14.5.68.4 Description

This cmdlet removes data perviously with `Set-KFSessionStorageData` persisted the the *kenaflow* session storage.

Please read the article about *Storage*.

By default `-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` are used - alone or in combination - to automatically add prefix(es) to your key.

14.5.68.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.68.6 Example

This writes and reads serialized data to the session storage.

```
    Set-KFSessionStorageData -Key "kenaflow" -Data @{"is" = "a workflow engine for␣
↪SharePoint, Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:
↪$false -AdditionalPassphare "cool"
    Write-KFLog (Get-KFSessionStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false -AdditionalPassphare "cool")
    Remove-KFSessionStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false
```

14.5.68.7 Related Cmdlets

- *Get-KFSessionStorageData*
- *Set-KFSessionStorageData*

14.5.69 Remove-KFStorageData

14.5.69.1 Syntax

```
    Remove-KFStorageData
      [-Key] <key>
      [-RelatedToWorkflow[:$true]]
      [-RelatedToItem[:$true]]
      [-ThrowErrors[:$true]]
```

14.5.69.2 Support

>= *kenaflow* 1.0.0

14.5.69.3 Return

Nothing.

14.5.69.4 Description

This cmdlet removes data perviously with *Set-KFStorageData* persisted the the *kenaflow* permanent storage.

Please read the article about *Storage*.

By default `-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` are used - alone or in combination - to automatically add prefix(es) to your key.

14.5.69.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.69.6 Example

This writes and reads serialized data to the permanent storage.

```
    Set-KFStorageData -Key "kenaflow" -Data @{"is" = "a workflow engine for␣
↪SharePoint, Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:
↪$false -AdditionalPassphare "cool"
    Write-KFLog (Get-KFStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false -AdditionalPassphare "cool")
    Remove-KFStorageData -Key "kenaflow" -RelatedToItem:$false -RelatedToWorkflow:
↪$false
```

14.5.69.7 Related Cmdlets

- *Set-KFStorageData*
- *Get-KFStorageData*

14.5.70 Reset-KFPermissionInheritance

14.5.70.1 Syntax

```
    Reset-KFPermissionInheritance
      [-Item <list-item>]
      [-ThrowErrors[:$true]]
```

14.5.70.2 Support

>= *kenaflow* 1.0.0

14.5.70.3 Return

Nothing.

14.5.70.4 Description

This command can be used to reset the inheritance of permissions on a list item if permission inheritance was stopped before.

The list element can be specified with parameter `-Item`. If it is not specified, the list element for which the workflow is currently being executed is used.

If permission inheritance was not stopped before the item permissions are not changed.

Obviously the cmdlet is only available for SharePoint workflows.

14.5.70.5 Parameters

`[-Item <list-item>]`

The SharePoint list item to be processed.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.70.6 Example

```
$myItem = Get-PnPListItem -List "Custom List" -Id 1
Stop-KFPermissionInheritance -Item $myItem
Reset-KFPermissionInheritance -Item $myItem
```

14.5.70.7 Related Cmdlets

- *Add-KFPermission*
- *Get-KFPermission*
- *Set-KFPermission*
- *Stop-KFPermissionInheritance*
- *Update-KFPermission*

14.5.71 Resolve-KFPrincipal

14.5.71.1 Syntax

```
Resolve-KFPrincipal
  -Principal <object>
  [-Item <item>]
  [-ClientContext <client-context>]
  [-AsObjects[:$false]]
  [-ThrowResolveErrors[:$false]]
  [-MaxLevel <int> (999999)]
  [-ResolveAtADLevel:[$true]]
  [-Flat:[$false]] [-Unique:[$false]]
  [-ThrowErrors[:$true]]
```

14.5.71.2 Support

>= *kenaflow* 2.0.38

14.5.71.3 Return

List of objects of type `kenaflow.Cmdlets.ResolveKFPrincipalResult`:

```
    namespace kenaflow.Cmdlets
    {
        public class ResolveKFPrincipalResult
        {
            public enum ResolveKFPrincipalResultObjectType {
                        Unknown,
                        SPGroup,
                            ADSecurityGroup,
                            ADDistributionGroup,
                            ADUser,
                            SPUserADUser,
                            SPUserADGroup
                    }

            public bool AD { get; internal set; }
            public ResolveKFPrincipalResultObjectType ObjectType { get; internal␣
↪set; }
            public string LoginName { get; internal set; }
            public string Title { get; internal set; }
            public string EMail { get; internal set; }
            public string ParentLogin { get; internal set; }
            public object Object { get; internal set; }
            public List<ResolveKFPrincipalResult> SubPrincipals { get; internal␣
↪set; }
        }
    }
```

14.5.71.4 Description

This cmdlet is only available in SharePoint workflows!

This cmdlet can be used to resolve all sorts of strings, including templates with placeholders, to resolve principals.

"Principals" are:

- SharePoint users objects
- SharePoint group objects
- Active Directory user objects
- Active Directory security group objects

The cmdlet is able to resolve all of them as long as it is able to connect to the active directory where the AD objects are is.

!! It is not possible currently to resolve Azure Active Directory objects in case of SharePoint Online. The cmdlet will only resolve and return SharePoint objects.

Example:

```
    Resolve-KFPrincipal -Principal "{{Author}}"
```

This resolves the user contained in the "Author" field of the current list item. (This will not work for *site* workflows except you specify a list item with parameter `-Item`.)

The *kenaflow* template engine is used to resolve the given strings. *Templates*

The "Author" will be resolved as a SharePoint user. But also - as "SubPrincipal" in the return object - as an Active Directory user.

If you specify `-Flat` the SharePoint user and its Active Directory user will return as a flat list at the same hierarchy level.... like `@( <sharepoint-user> , <ad-user> )` (pseudo code!).

If you specify `-ResolveAtADLevel:$false` Active Directory objects are not resolved. In the example you would only get the SharePoint user object.

Another Example:

You have a custom list with a field "Responsible" of type "Person" with setting "Person or Group". - You have a list item with value "Intranet Members" in the field. - "Intranet Members" is a default SharePoint group on you site. It contains some SharePoint users, some AD security groups and some SharePoint groups.

The cmdlet would resolve all of them with its default settings. These calls are the same:

- `Resolve-KFPrincipal -Principal $item["Responsible"]`
- `Resolve-KFPrincipal -Principal @($item["Responsible"])`
- `Resolve-KFPrincipal -Principal "{{Responsible}}"`
- `Resolve-KFPrincipal -Principal @("{{Responsible}}")`

First "Responsible" is resolved as SharePoint group object. All members of this group are analysed. That could be SharePoint users and AD security groups. (A SharePoint group cannot contain other SharePoint groups.) All AD security groups are resolved. If they contain other AD security groups they are also resolved. And so on. To the last level where all group members are AD users. - You can specify `-MaxLevel` to stop the resolving at the depth indicated. - - The return object is hierarchical. That means that you can see the resolved structure in the return object and go into deep using property `SubPrincipals`.

You can - as described before - use `-Flat` to receive a flat list of all objects. If using `-Flat` you can additionally use `-Unique` to remove duplicate objects from the return object.

A word to `-ThrowResolveErrors`: SharePoint user objects "normally" belong to existing Active Directory objects. But in AD the can be deleted. They are not deleted in SharePoint too because list items are referencing to them, e.g. in the fields "Author" or "Editor". If *kenaflow* tries to resolve this principals it will get an error because the objects are invalid or the corresponding AD user objects do not exist. In this case the cmdlet will "silently continue" if the `-ThrowResolveErrors` is **not** specified!

The cmdlet can also be called with the name(s) of a user(s) or group(s) as string, e.g.:

- `Resolve-KFPrincipal -Principal "Intranet Members"`
- `Resolve-KFPrincipal -Principal "Intranet Members;Intranet Visitors"`
- `Resolve-KFPrincipal -Principal "Intranet Members,Intranet Visitors"`
- `Resolve-KFPrincipal -Principal @("Intranet Members")`
- `Resolve-KFPrincipal -Principal @("Intranet Members","Intranet Visitors")`

You can combine it with template processing:`Resolve-KFPrincipal -Principal "Intranet Members,{{Author}};{{Editor}};ingo@sharepoint.farm"`which is the same as`Resolve-KFPrincipal -Principal @("Intranet Members","{{Author}}"; $item["Editor"];"ingo@sharepoint.farm")`

14.5.71.5 Parameters

`-Principal <object>`

The object(s) (list, string, SharePoint User Field value, SharePoint User Field value array) that are to be resolved.

`[-Item <item>]`

The SharePoint list item that is used for template processing (e.g. "{{Author}}")

`[-ClientContext <client-context>]`

A SharePoint client context is necessary to resolve SharePoint objects. If nothing is specified the client context is used from the given item (`-Item`) or from the current processed list item. - But if you need for some reasons to specify your own client context you can use this parameter.

`[-AsObjects[:$false]]`

You will not receive a return object of type `kenaflow.Cmdlets.ResolveKFPrincipalResult` but as raw Active Directory object or SharePoint client object model objects. - Active Directory objects are comming from .NET namespace System.DirectoryServices.AccountManagement.

`[-ThrowResolveErrors[:$false]]`

If the resolve of a given string or whatever failed not the complete execution of the cmdlet will fail. You get the found pricinals in the return object.

`[-MaxLevel <int> (999999)]`

Specifies the depth of the SharePoint Group and Active Directory analysis. - You could specify `1` to just resolve the very first level.

`[-ResolveAtADLevel:[$true]]`

If set to `$false` it will not resolve at Active Directory level.

`[-Flat:[$false]]`

If specified you get a flat list not a hierarchical tree.

`[-Unique:[$false]]`

In case of a `-Flat` list you get resolved objects only once.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.71.6 Example

Have a look at the section "Description" above.

14.5.72 Send-KFMail

14.5.72.1 Syntax

```
Send-KFMail
  [-RedirectMailMessage <mail-message>]
  [-Body] <string>
  [-Subject] <string>
  [-To] <address>
  [-CC <address>]
```

(continues on next page)

(continued from previous page)

```
    [-BCC <address>]
    [-From <address>]
    [-ReplyTo <address>]
    [-Attachments <object-or-list>]
    [-IsHTMLMail[:$true]]
    [-ThrowResolveErrors[:$false]]
    [-FloodProtectionFile <filename>]
    [-FloodProtectionTBE <time-in-seconds>]

    [-Item <item-object>]
    [-ClientContext <client-context-object>]
    [-IgnoreSharePointGroups[:$true]]
    [-IgnoreUserWithEmptyMailAddress[:$true]]
    [-ResolveSharePointGroups[:$false]]
```

14.5.72.2 Support

>= *kenaflow* 1.0.0

"LinkedResources" in Attachments are supported starting with *kenaflow 2.0.33*.

14.5.72.3 Return

Nothing

14.5.72.4 Description

With this cmdlet you can easily send mails.

SMTP-Server, Credentials, Port, etc. are configured in *kenaflow* global configuration, section *"Mail"*.

To prevent mail flood you can specify a flood protection file that contains the timestamp of last send mail. If you specify this together with `-FloodProtectionTBE` mails will not be send for the specified amount of time for all `Send-KFMail` statements that use the same flood protection file.

Normally the parameters `-To`, `-CC` and `-BCC` accept strings with `,` or `;` separated lists of email addresses or SharePoint lists with such strings, like ` @("test1@example.com", "test2@example.com;test3.example.com").

`-ReplyTo` and `-From` are taken from the workflow config `_wfconfig.ps1` or - if not specified there - from the global configuration of kenaflow.

If you use the cmdlet in a SharePoint workflow script you can use some additional parameters to control the behaviour. Especially the "resolving tool" for email addresses are more advanced than for the other workflows because addressees are collected from different sources. Also templates with placeholders are available for addressing

Example: `-To "{{Author}};{{Editor}}"` This can be used to address the creator and the last editor of the currently processed list item. The *kenaflow* template engine is used to resolve the strings: *Templates*.

The result of resolving the address strings of `-To`, `-CC` and `-BCC` are lists of email addresses. If there goes something wrong to determine email addresses the cmdlet will fail with an exception.

To reduce the cases of errors you can specify `-IgnoreUserWithEmptyMailAddress`.

SharePoint groups are also a problem because it's not possible by default to send their users a mail. You can specify `-IgnoreSharePointGroups` to ignore all SharePoint groups that are resolved. Or you specify `-ResolveSharePointGroups` to ask *kenaflow* to resolve SharePoint groups and use their members as recipients.

If there are exceptions they will not be thrown as long as you do not use `-ThrowResolveErrors`.

If the template placeholders like `{{Author}}` should be taken from a different list item or in a *site* workflow where you do not have a "currently processed" list item by default, you can use `-Item` to specify a list item.

You can use parameter `-RedirectMailMessage` in *email* workflows to re-send the currently processed mail. It is send by *kenaflow* with its SMTP settings. The `<mail-message>` object (as described in this article: *Email Workflows*) must be provided.

The following properties of the `<mail-message>` are used. They can be overwritten by specifying the corresponding cmdlet parameters.

- `Body`
- `Subject`
- `Attachments`

To use the `HtmlBody` of the `<mail-message>` please specify `-IsHtmlMail` with the cmdlet.

! If a prefix and/or suffix for email is configured for the *kenaflow* instance (Global Config section *"Mail"*) this/these is/are added to the subject of the re-send email!

See the example below.

14.5.72.4.1 Attachments

You can attach files to the mail.

You can specify an object like this: `@{FileName = "..."; <source> = <object>; ContentType = "..."; ContentId = "..."}`

Since *kenaflow 2.0.33* you can specify `LinkedResource = $true` in the attachment so configure the attachment to be a "LinkedResource", e.g. for embedded images. Please see the examples below. In this case you can omit the `FileName`. LinkedResources only work if `-IsHTMLMail` is specified.

`ContentType` and `ContentId` are optional in case of `LinkedResource = $false` (default if not specified) but required for "LinkedResource".

You can eigther specify one attachment object or a list of such object with `@( @{...}, @{...}, ... )`

Description of `<source> = <object>` :

`<soure>` can be one of this keys:

- `URL` : is a server-relative Url in SharePoint. **If** the referenced file is in another web you can add property `ClientContext` to the attachment object.
- `Stream` : is an open stream object
- `Base64` : is a base64 encoded string that represents the attachment.
- `Path` : is a path to a file. Paths can be relative to the workflow folder.
- `Bytes` : is a attachment given as Byte array.
- `Text` : is a plain text.

`ContentType` can be specified as full MIME content type like "image/jpeg" or as one of these shortcuts:

- PNG
- TIFF or TIF
- JPEG or JPG
- PDF

- SVG

If the shortcut is other than this the content type "application/octet-stream" will be used for "unspecified type such as binary".

14.5.72.4.2 Best Practice Pattern

Ideally, you define the template in the configuration list and use it directly in the mail via KFTemplate. It is also recommended to start the subject in the mail with a unique identifier for each workflow so that the user can assign it more easily.

```
    $body = (Invoke-KFTemplate -Template $config['tpl.Mailtemplate'])

    Send-KFMail -To "{{Editor}}" -CC "{{Author}}" -IsHTMLMail -Subject
↪"Workflowname: Item with ID {{ID}} changed" -Body $body
```

14.5.72.5 Parameter

`[-RedirectMailMessage <mail-message>]`

You can re-send mails from *email* workflows. Please read the description above.

`[-Body] <string>`

The mail body.

`[-Subject] <string>`

The mail subject.

`[-To] <address(es)>`

The receiver of the mail. This string is automatically processed by the *kenaflow* template engine. You can specify SharePoint item fields like this: `-To "{{Editor}};{{Author}}"` This will send the mail to the creator and the last editor of the current SharePoint list item or of the specified list item in `-Item`. You can specify strings from the workflow configuration list like this: `-To "[[some_config_key]]"`

`[-CC <address(es)>]`

CC receiver. The string is processed by the *kenaflow* template engine. Please see description of parameter `-To` above.

`[-BCC <address(es)>]`

Bcc receiver. The string is processed by the *kenaflow* template engine. Please see description of parameter `-To` above.

`[-From <address>]`

This can be specified to overwrite the sender address configured in `_wfconfig.ps1` OR in Global Configuration (if not overwritten by `_wfconfig.ps1`).

`[-ReplyTo <address>]`

This can be specified to overwrite the reply-to address configured in `_wfconfig.ps1` OR in Global Configuration (if not overwritten by `_wfconfig.ps1`).

`[-Attachments <object-or-list>]`

Please read the section "Attachments" above.

`[-IsHTMLMail[:$true]]`

By default mail is send as plain text. But if you specify this switch parameter the mail is send as HTML mail. You need to specify HTML code in the `-Body` by yourself.

`[-ThrowResolveErrors[:$false]]`

By default no address resolving errors are thrown. Not resolved users/placeholders/... are ignored. The mail is only send to receipients whos email address could be resolved.

`[-FloodProtectionFile <filename>]`

This setting together with `-FloodProtectionTBE` can be used to configure flood protection. If specified the timestamp of the last mail ist stored in this file. If a relative path is used it's relative to the `_wfdata` folder inside the workflow folder.

`[-FloodProtectionTBE <time-in-seconds>]`

This setting together with `-FloodProtectionFile` can be used to configure flood protection. Here you specify the amout of time (in seconds) that need to elapse before a new mail will be send. This time is relative to the timestamp stored in the flood protection file.

`[-Item <item-object>]`

Here you can specify a different list item if you want to use the template engine for `-To`, `-CC` and `-BCC` with values from another list item than the current.

This setting is only available in SharePoint related workflows.

`[-ClientContext <client-context-object>]  [-IgnoreSharePointGroups[:$true]]`
`[-IgnoreUserWithEmptyMailAddress[:$true]] [-ResolveSharePointGroups[:$false]]`

14.5.72.6 Examples

In this example, the mail is sent to the person who created and changed an entry and the ID of the changed element is specified in the mail.

```
    Send-KFMail -To "{{Editor}}" -CC "{{Author}}" -IsHTMLMail `
      -Subject "Something changed" -Body "ItemID: {{ID}}" `
      -From "{{Editor}}" -ReplyTo "[[user.workflowowner]]"
```

In the following example you see how to send a plain text string as mail attachment. The string array for the `-Body` is converted to a single string. This is required and the code snipped can be useful sometimes.

```
    $lines = @()
    $lines += "An email"
    $lines += "from kenaflow"

    Send-KFMail -To "somebody@example.com" -Subject "Mail from kenaflow" `
      -Body ([string]::Join(" ", $lines)) `
      -Attachments @(
         @{
              FileName="PlainText.txt";
              Text="This is a test from kenaflow";
              ContentType="text/plain"
         }
       )
```

In this example a mail with an embedded image as "LinkedResource" is send to the person stored in custom field "Assignee". The embedded image is stored in the workflow folder.

```
    Send-KFMail -Subject "A mail send with kenaflow" -To "{{Assignee}}" -Body "
↪<html><body><img src='cid:myImg'></body></html>" -Attachments @(
            @{ Path = "kenaflow.png"; ContentType="png"; ContentId = "myImg";␣
↪LinkedResource = $true}
    )
```

In this example a mail with an embedded image as "LinkedResource" is send to the person stored in custom field "Assignee". The embedded image is stored in SharePoint.

```
    Send-KFMail -Subject "A mail send with kenaflow" -To "{{Assignee}}" -Body "
↪<html><body><img src='cid:myImg'></body></html>" -Attachments @(
            #server relative URL!!
            @{ URL = "/sites/kenaflow/SiteAssets/kenaflow.png"; ContentType="png";␣
↪ContentId = "myImg"; LinkedResource = $true}
    )
```

In the following case we use a file from another site collection

```
    $otherSiteCollectionContext = new-object Microsoft.sharepoint.client.
↪clientcontext("https://sharepoint.farm/sites/kenaflow")

    send-kfmail -subject "test" -to "ik@kenaro.com" -body "<html><body><img src=
↪'cid:myImg'></body></html>" -attachments @(
            #server relative URL!!
            @{ URL = "/sites/kenaflow/shared documents/test.png"; ContentType="png
↪"; ContentId = "myImg"; LinkedResource = $true; ClientContext=
↪$otherSiteCollectionContext}
    )
```

This will redirect an email from a email workflow script. “From” (Sender) and “To” (Recipient) are taken from the cmdlet parameters. All other email properties (Subject, Body, Attachments) are taken from the processed email. `HtmlBody` is used from the processed email if present.

```
    Send-KFMail -RedirectMailMessage $mail -From "kenaflow@example.com" -To
↪"me@example.com" -IsHtmlMail
```

14.5.72.7 Related Cmdlets

- *Invoke-KFTemplate*

14.5.73 Set-KFApproval

14.5.73.1 Syntax

```
    Set-KFApproval
      [-ModerationState] <moderation-state> (Approved)
      [-Item <list-item>]
      [-Editor <sharepoint-user-object> ($null)]
      [-ThrowErrors[:$true]]
```

14.5.73.2 Support

>= *kenaflow* 1.0.0

14.5.73.3 Return

Nothing.

14.5.73.4 Description

This cmdlet is only available in SharePoint workflows!

It is used to immediately set the moderation state of the currently processed item or the item specified with `-Item`.

You need to known that this setting of the approval state will not last long because after further modifications of the item the moderation state is automatically reset by SharePoint.

To set the "final moderation state" at the end of the item processing please use cmdlet *Set-KFFinalModerationState*. This will be processed after *kenaflow* is (almost) finished with the current item.

When calling this cmdlet item modifications are send immediately to SharePoint.

14.5.73.5 Parameters

`[-ModerationState] <moderation-state> (Approved)`

The desired moderation state. Valid values are:

- `Approved`
- `Rejected`
- `Pending`
- `Draft`
- `Scheduled`

`[-Item <list-item>]`

The list item to be used. If nothing is specified the currently processed list item is used.

`[-Editor <sharepoint-user-object> ($null)]`

The "editor" user to be set.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.73.6 Example

```
Set-ModerationState Approved
```

14.5.73.7 Related Cmdlets

- *Set-KFFinalModerationState*
- *Set-KFFinalCheckin*
- *Set-KFItemModifiedFlag*

14.5.74 Set-KFContext

```
## empty
```

14.5.75 Set-KFData

14.5.75.1 Syntax

```
    Set-KFData
      [-Key] <string>
      [-Data] <object>
      [-ThrowErrors[:$true]]
```

14.5.75.2 Support

>= *kenaflow* 1.0.0

14.5.75.3 Return

Nothing.

14.5.75.4 Description

This cmdlet writes the data as serialized string (CliXML format and JSON format) to the SharePoint workflow *data list*. (The actual name of the list is specified in the workflow configuration file `_wfconfig.ps1`.)

Serialization does not allow the entire state of a complex object to be transferred in CliXML or JSON, but ultimately only the data of the objects. This must be taken into account. For example, a SharePoint list element cannot be serialized as an object (ListItem object) so that it could be reloaded and modified later. This limitation is a typical problem in software development and not limited to *kenaflow*.

If an object was persisted before with the same key it gets overwritten without warning!

14.5.75.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-Data] <object>`

The object that is to be persisted.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.75.6 Example

```
    Set-KFData -Key "kenaflow" -Data @{"is" = "a workflow engine for SharePoint,␣
↪Email and PowerShell"}
    Write-KFLog (Get-KFData -Key "kenaflow")
    Remove-KFData -Key "kenaflow"
```

14.5.75.7 Related Cmdlets

- *Get-KFData*
- *Remove-KFData*

14.5.76 Set-KFDateTime

14.5.76.1 Syntax

```
    Set-KFDateTime
      [-ReferenceDateTime] <datetime>
      [-UTC[:$false]]
```

```
    Set-KFDateTime
      [-AddSeconds] <int>
      [-ThrowErrors[:$true]]
```

14.5.76.2 Support

>= *kenaflow* 2.0.0

14.5.76.3 Return

Nothing.

14.5.76.4 Description

This is a tool cmdlet for debugging.

Together with *Get-KFDateTime* and *Set-KFDateTime* this cmdlet can be used to simulate time differences in a debugging context. First you set a time difference with `Set-KFDateTime` that is relative to the current time. When you get the time using `Get-KFDateTime` the difference is added or substracted from the current time. In combination with *Get-KFInDebug* you can simulate a workflow run "later" or "earlier", e.g. if you check time stamps against the current time.

With `Set-KFDataTime` you set the time difference.

The *first option* is to set a difference based on a DateTime specified with `-ReferenceDateTime <datetime>`. During execution the difference between this DateTime and `DateTime.Now` is calculated. When you later get the DateTime with `Get-KFDateTime` this calculated difference is used to manipulate the current datetime. - When using `-ReferenceDateTime` together with `-UTC` the given DateTime is handled as UTC and the difference is calculated agains `DateTime.UtcNow`.

The *second option* is to set a difference based on an amount of seconds specified with `-AddSeconds <int>`. This can also be a negative value. When you later get the DateTime with `Get-KFDateTime` this amount of seconds is added to manipulate the current datetime.

14.5.76.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.76.6 Example

```
Clear-KFDataTime
```

14.5.76.7 Related Cmdlets

- *Get-KFDateTime*
- *Clear-KFDataTime*
- *Get-KFInDebug*

14.5.77 Set-KFDBData

14.5.77.1 Syntax

```
Set-KFDBData
  [-Key] <key>
  [-Data] <data>
  [-RelatedToWorkflow[:$true]]
  [-RelatedToItem[:$true]]
  [-ThrowErrors[:$true]]
```

14.5.77.2 Support

>= *kenaflow* 1.0.0

14.5.77.3 Return

Deserialized object or `$null`.

14.5.77.4 Description

This cmdlet serialize and persist data to the *kenaflow* SQLite database.

If the requested object exist, it will be overwritten. An exception is not thrown.

We have described this in article *Persisting Data*, section "*kenaflow* SQLite Persistence".

By default `-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` are used - alone or in combination - to automatically add prefix(es) to your key.

14.5.77.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-Data] <data>`

This is the data to be serialized and persisted.

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.77.6 Example

```
    Set-KFDBData -Key "kenaflow" -Data @{"is" = "a workflow engine for SharePoint,␣
↪Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:$false
    Write-KFLog (Get-KFDBData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false)
    Remove-KFDBData -Key "kenaflow" -RelatedToItem:$false -RelatedToWorkflow:$false
```

14.5.77.7 Related Cmdlets

- *Get-KFDBData*
- *Remove-KFDBData*

14.5.78 Set-KFFinalAuthor

14.5.78.1 Syntax

```
    Set-KFFinalAuthor
      [-User] <string-or-object>
      [-ThrowErrors[:$true]]
```

14.5.78.2 Support

>= *kenaflow* 2.0.47

14.5.78.3 Return

Nothing.

14.5.78.4 Description

With this cmdlet you can specify a SharePoint user that will be set as "Author" of the list item or file at the end of the processing of the current item.

It is only available in SharePoint *list* and *state machine* workflows with item context.

If you use this cmdlet you can force a user object to be set.

You can use it multiple times in a workflow script. The calls replace the previoius set values.

If you set `$null` the "Author" field is not changed.

14.5.78.5 Parameters

`[-User] <string-or-object>`

A string that gets resolved to exactly 1 user object. Or a user object.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.78.6 Example

```
Set-KFFinalAuthor -User "{{Editor}}"
```

14.5.78.7 Related Cmdlets

- *Set-KFFinalEditor*

14.5.79 Set-KFFinalCheckin

14.5.79.1 Syntax

```
Set-KFFinalCheckin
  [-Checkin] <checkin-type>
  [-Comment] <comment>
  [-ThrowErrors[:$true]]
```

14.5.79.2 Support

>= *kenaflow* 1.0.0

14.5.79.3 Return

Nothing.

14.5.79.4 Description

This cmdlet is used to set the final checkin operation after *kenaflow* finished processing of the current list item.

This cmdlet is only available in SharePoint *list* and *state machine* workflows.

The specified `-Checkin` is only used for list items or files if the list or library has a setting "force checkout". In this case the list item or file is checked out before processing. Of cause is has to be checked in at the end of processing.

With this cmdlet you can set the checkin options: Should a major version be published or a minor version? What is the comment for the checkin?

You can use the cmdlet multiple times in the workflow script, but only the last set values are used.

If it is not used at all a "minor" version is checked in with no comment.

14.5.79.5 Parameters

`[-Checkin] <checkin-type>`

The checkin type. Valid values:

- `Major` : checkin as major version
- `Minor` : checkin as minor version
- `None` : same as `Minor`

`[-Comment] <comment>`

The checkin comment.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.79.6 Example

```
Set-KFFinalCheckin Major "published by kenaflow!"
```

14.5.79.7 Related Cmdlets

- *Set-KFApproval*
- *Set-KFFinalModerationState*
- *Set-KFItemModifiedFlag*

14.5.80 Set-KFFinalEditor

14.5.80.1 Syntax

```
Set-KFFinalEditor
  [-User] <string-or-object>
  [-ThrowErrors[:$true]]
```

14.5.80.2 Support

>= *kenaflow* 2.0.47

14.5.80.3 Return

Nothing.

14.5.80.4 Description

It is only available in SharePoint *list* and *state machine* workflows with item context.

With this cmdlet you can specify a SharePoint user that will be set as "Editor" of the list item or file at the end of the processing of the current item.

Normally *kenaflow* will set the "last editor" at the end of the item processing to be as transparent as possible to the users. The "last editor" normally is a person that edits the list item. After processing the item by a workflow it would have the *kenaflow* service account as "last editor". So *kenaflow* re-sets the "last editor" with the last known one.

If you use this cmdlet you can force a user object to be set.

You can use it multiple times in a workflow script. The calls replace the previoius set values.

If you set `$null` the original "last editor" is set!

14.5.80.5 Parameters

`[-User] <string-or-object>`

A string that gets resolved to exactly 1 user object. Or a user object.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.80.6 Example

```
Set-KFFinalAuthor -User "{{Editor}}"
```

14.5.80.7 Related Cmdlets

- *Set-KFFinalAuthor*

14.5.81 Set-KFFinalModerationState

14.5.81.1 Syntax

```
Set-KFFinalModerationState
  [-ModerationState] <moderation-state>
  [-ThrowErrors[:$true]]
```

14.5.81.2 Support

>= *kenaflow* 1.0.0

14.5.81.3 Return

Nothing.

14.5.81.4 Description

This cmdlet is used to set the final moderation state **after** *kenaflow* finished processing of the current list item.

This cmdlet is only available in SharePoint *list* and *state machine* workflows.

The moderation state is only relevant to lists or libraries that have "content approval" enabled. If so the "view only" users will not see unpublished and/or not-approved (rejected) items or files.

You can use the cmdlet multiple times in the workflow script, but only the last set values are used.

When a value other than `DoNotSet` is specified the moderation state will be written to the currently processed list item.

!! In case of setting the moderation state it is not possible to rewrite the last "Editor" and last "Modified" timespamp. The list item will be - for example - approved but with the name of the workflow execution account, normally the *kenaflow* service account.

14.5.81.5 Parameters

`[-ModerationState] <moderation-state>`

The moderation state. Valid values:

- `DoNotSet` : The moderation state will NOT be set at the end of the current item processing.
- `Approved`
- `Rejected`
- `Pending`
- `Draft`
- `Scheduled`

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.81.6 Example

```
Set-KFFinalModerationState Approved
```

14.5.81.7 Related Cmdlets

- *Set-KFApproval*
- *Set-KFFinalCheckin*
- *Set-KFItemModifiedFlag*

14.5.82 Set-KFItemData

14.5.82.1 Syntax

```
Set-KFItemData
  [[-Key] <string>]
  [-Data] <object>
  [-Item <list-item-object>]
  [-ThrowErrors[:$true]]
```

14.5.82.2 Support

>= *kenaflow* 1.0.0

14.5.82.3 Return

Nothing.

14.5.82.4 Description

!! This is only available in SharePoint *list* or *state machine* workflows!

The cmdlet can be used to store data in the context of the list item that can be reloaded later.

The data is serialized using the specified `-Key` and the unique ID of the currently processed list item in the workflow *data list*.

Serialization means that the objects are brought into a storable format, here PowerShell CLIXML and JSON.

To store several different objects in the context of the list element, the additional `-Key` can be specified.

Serialization does not allow the entire state of a complex object to be transferred in CliXML or JSON, but ultimately only the data of the object. This must be taken into account. For example, a SharePoint list element cannot be serialized as an object (ListItem object) so that it could be reloaded and modified later. This limitation is a typical problem in software development and not limited to *kenaflow*.

In the workflow data list, a persistence key is formed from the workflow ID, the list element ID and, if specified, the `-Key`. This links the data to the workflow and to the list element.

To get the data, use the *Get-KFItemData* cmdlet.

14.5.82.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-Data] <object>`

The data that is to be persisted.

`[-Item <list-item-object>]`

The SharePoint list item in whose context the data is serialized.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.82.6 Example

```
    $item = Get-PnPListItem -List "Custom List" -Id 1
    Set-KFItemData -Item $item -Key "kenaflow" -Data @{"is" = "a workflow engine␣
↪for SharePoint, Email and PowerShell"}
    Write-KFLog (Get-KFItemData -Item $item -Key "kenaflow")
    Remove-KFItemData -Item $item -Key "kenaflow"
```

14.5.82.7 Related Cmdlets

- *Get-KFItemData*
- *Remove-KFItemData*

14.5.83 Set-KFItemModifiedFlag

14.5.83.1 Syntax

```
    Set-KFItemModifiedFlag
      [-ThrowErrors[:$true]]
```

14.5.83.2 Support

>= *kenaflow* 1.0.0

14.5.83.3 Return

Nothing.

14.5.83.4 Description

This is used to force *the kenaflow field* field to be updated with the current workflow run timestamp.

14.5.83.5 Parameters

`[-ThrowErrors[:$true]]`

> This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.83.6 Example

```
Set-KFItemModifiedFlag
```

14.5.83.7 Related Cmdlets

- *Set-KFApproval*
- *Set-KFFinalModerationState*
- *Set-KFFinalCheckin*

14.5.84 Set-KFObject

14.5.84.1 Syntax

```
Set-KFObject
  -Name <string>
  -Object <object-as-PSObject>
  [-ThrowErrors[:$true]]
```

14.5.84.2 Support

>= *kenaflow* 4.0.11

14.5.84.3 Return

14.5.84.4 Description

This cmdlet can be used to store data in an in-memory-cache during the **current** workflow execution. After the workflow processing is finished the in-memory-cache is cleared. The workflow script has no access to the in-memory-cache of other workflows.

If this cmdlet is used in a workflow script that is executed from an remote event the in-memory-cache is cleared after the current event processing.

14.5.84.5 Parameters

`[-Name] <string>`

The key that is used to write data

`[-Object] <object-as-PSObject>`

This is the data to be stored in memory.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.84.6 Example

The `proProcessingScript` specified in `_wfconfig.ps1` could include the following line of Powershell script code:

```
    Set-KFObject -Name "ListOfDataForExcelExport" -Object @()
```

The SharePoint workflow script could include this script code:

```
    $d = Get-KFObject -Name "ListOfDataForExcelExport"
    $d += @{Title = $item["Title"]; Id = $item.Id; Author = $item["Author"].
↪LookupValue; Editor = $item["Editor"].LookupValue}
    Set-KFObject -Name "ListOfDataForExcelExport" -Object $d
```

This creates an empty list in memory.

The `postProcessingScript` specified in `_wfconfig.ps1` could include the following line of Powershell script code:

```
    $data = Get-KFObject -Name "ListOfDataForExcelExport"
    New-KFExcelFile -Input $data -SheetName "Output" -AsBase64 -Fields @("Id",
↪"Title", "Author", "Editor")

    Remove-KFObject -Name "ListOfDataForExcelExport" # This is not required␣
↪because at the end of a workflow processing the in-memory-object is cleared!
```

14.5.84.7 Related Cmdlets

- *Get-KFObject*
- *Remove-KFObject*

14.5.85 Set-KFPermission

14.5.85.1 Syntax

```
    Set-KFPermission
      [-Permissions] <list-of-permissions>
      [-Item <list-item>]
      [-CopyCurrentPermissions[:$false]]
      [-ThrowResolveErrors[:$false]]
      [-ThrowErrors[:$true]]
```

14.5.85.2 Support

>= *kenaflow* 1.0.0

14.5.85.3 Return

Nothing.

14.5.85.4 Description

Use this cmdlet to add permissions to a list item.

Set means that it **replaces** permissions! If you want to add some permissions to the current permissions please use cmdlet *Add-KFPermission*.

The `-Permissons` parameter gets a single permission object or a list of permission objects of the following form.

The permission object is specified as a PowerShell Hashtable:

```
@{
   User = <string-or-object>;
   PermissionsSets = @(<permission-set-name>[, <permission-set-name>[, ...]]);
   Quick = "CRUDMA";
)
```

The `User` property of the permission object can be:

- List of strings or objects
- string, that describe the user
- object as (array of) SharePoint `FieldUserValue`

Such a "string" of the `User` property of the permission object can be:

- a user login name
- an email address
- a SharePoint field of the currently processed or with `-Item` specified list item
- a combination of all of them separated by "," or ";"

Examples:

- `User = @($item["Author"])`
- `User = $item["Author"]`
- `User = @($item["Author"], $item["Editor"])`
- `User = "{{Author}}"`
- `User = "{{Author}},{{Editor}}"`
- `User = @($item["Author"], "{{Editor}},sharepoint\ingo", "ingo@sharepoint.farm")`

The *kenaflow* template engine is used to resolve the given strings. *Templates*

Also SharePoint groups are resolved. E.g. `User = @("Intranet Owners")`

The parameter `-PermissionSets` is a PowerShell list of one or more names of valid / existing SharePoint permission sets, such as "Full Control" or "Design". - Custom permission sets can be named too. But the have to exist!

The parameter `-Quick` is a kind of shortcut to `-PermissionSets`. *kenaflow can* create it's own permission sets as specified in the `_wfconfig.ps1` file. They have to be created using program switch `--createpermissionsets`. This permission sets are by default named as:

- kenaflow Create => `C`
- kenaflow Read => `R`
- kenaflow Update => `U`
- kenaflow Delete => `D`
- kenaflow Manage => `M`
- kenaflow Approve => `A`

The first letter(s) **in combination** can be used in parameter `-Quick` to specify these permission sets. - But they have to be configured in `_wfconfig.ps1` and must exist in SharePoint! - For example: `Quick="CUD"` => "Create", "Update", "Delete" - but not "Read".

The base permissions of each of the *kenaflow* defined permission sets can be found here: *Permissions*.

This is an example for a valid permission object - if the *kenaflow* permission sets are configured.

```
@{
    User = @("{{Author}}", $item["Editor"]);
    PermissionSets = @("Read");
    Quick = "CUD"
}
```

Multiple permission objects can be specified as a list:

```
@(
    @{User="sharepoint\ingo"; Quick="CUD"},
    @{User="sharepoint\kenaflow";PermissionSets=@("Full Control")}
)
```

14.5.85.5 Parameters

`[-Permissions] <list-of-permissions>`

One or more permission objects as described above.

`[-Item <list-item>]`

The list item to be processed.

`[-CopyCurrentPermissions[:$false]]`

If a list item with active permission inheritance is processed the permission inheritance will be stopped. In this case - and if this parameter is specified - the parents permissions are copied to the item before setting the correct permissions. If the parameter is not specified the permissions are blank after stopping permission inheritance.

`[-ThrowResolveErrors[:$false]]`

kenaflow tries to resolve the given `User` property of each permission object. If this fails and `-ThrowResolveErrors` is specified the cmdlet will fail. Otherwise the not existing or not successfully resolved principal is skipped.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.85.6 Examples

```
Set-KFPermission -Permissions @(
    @{
        User = "user1@sharepoint.farm";
        PermissionSets = @("Full Control", "Contribute");
        Quick="CRUDMA";
    },
    @{
        User = "user2@sharepoint.farm";
        PermissionSets = "Read";
        Quick="CRUDMA";
    }
)
```

```
$newEditor = "sharepoint\user3"

Set-KFPermission -Permissions @(
    @{ User="{{Author}}"; Quick="RM" },
    @{ User="{{currentEditor}}"; Quick="RM" },
    @{ User=$newEditor; Quick="RM" }
)
```

14.5.85.7 Related Cmdlets

- *Add-KFPermission*
- *Get-KFPermission*
- *Reset-KFPermissionInheritance*
- *Stop-KFPermissionInheritance*
- *Update-KFPermission*

14.5.86 Set-KFScriptRunTimeout

14.5.86.1 Syntax

```
Set-KFScriptRunTimeout
  -AdditionalScriptTimeout <additional-script-timeout-in-seconds>
```

```
Set-KFScriptRunTimeout
  -TotalScriptTimeout <total-script-timeout-in-seconds>
```

14.5.86.2 Support

>= *kenaflow* 2.0.18

14.5.86.3 Description

Workflow scripts are executed inside PowerShell. As in every programming language it is possible in PowerShell to create loops. It is not possible for *kenaflow* to check the "code quality" of a workflow script. If you (accidentally) create an endless loop *kenaflow* would not be able to stop the script. Therefore it has an security feature that forces scripts to stop if they run too long.

The initial setting for maximum script runtime is set in Global Configuration setting `defaultScriptRunTimeout` or in the workflow configuration ***missing***.

If a workflow knows that it needs more time the execution timeout can be set inside the workflow script during execution.

There are two options:

- Add some time to the current timeout.
- Set the total execution timeout.

The total or remaining timeout can be read with cmdlet `Get-KFScriptRunTimeout`

14.5.86.4 Parameter

`-AdditionalScriptTimeout <additional-script-timeout-in-seconds>`

Add the `<additional-script-timeout-in-seconds>` specified amount of seconds to the total script run timeout.

`-TotalScriptTimeout <total-script-timeout-in-seconds>`

Set the total script timeout in seconds.

14.5.86.5 Examples

```
***missing***
```

14.5.87 Set-KFSessionStorageData

14.5.87.1 Syntax

```
Set-KFStorageData
  [-Key] <key>
  [-Data] <data>
  [-Binary[:$false]]
  [-AdditionalPassphase <string>]
  [-RelatedToWorkflow[:$true]]
  [-RelatedToItem[:$true]]
  [-ThrowErrors[:$true]]
```

14.5.87.2 Support

>= *kenaflow* 1.0.0

14.5.87.3 Return

Nothing.

14.5.87.4 Description

This cmdlet can be used to write data to the *kenaflow* session storage.

Please read the article about *Storage*.

You can serialize objects using the PowerShell serializer when **NOT** specifying `-Binary`.

When using `-Binary` you must specify an byte array (`System.Byte[]`) with `-Data`.

If the requested object exist, it will be overwritten. An exception is not thrown.

By default `-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` are used - alone or in combination - to automatically add prefix(es) to your key.

14.5.87.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-Data] <data>`

This is the data to be serialized and persisted.

`[-Binary:[$false]]`

If NOT specified the cmdlet will store the given object as serialized string. This requires that data for the given `Key` will NOT be read with `-Binary` with `Get-KFStorageData`. If specified the cmdlet requires an byte array (`System.Byte[]`) with parameter `-Data`. The binary data is written to the storage. - This requires that data for the given `Key` will be read with `-Binary` with `Get-KFStorageData`.

`[-AdditionalPassPhrase <passphrase>]`

Workflow developers can specify an additional encryption passphrase. This needs to be specified for retrieving the data with `Get-KFStorageData`!

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.87.6 Example

This writes and reads serialized data to the session storage.

```
    Set-KFSessionStorageData -Key "kenaflow" -Data @{"is" = "a workflow engine for␣
↪SharePoint, Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:
↪$false -AdditionalPassphare "cool"
    Write-KFLog (Get-KFSessionStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false -AdditionalPassphare "cool")
    Remove-KFSessionStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false
```

14.5.87.7 Related Cmdlets

- *Get-KFSessionStorageData*
- *Remove-KFSessionStorageData*

14.5.88 Set-KFSPDatetimeLocal

14.5.88.1 Syntax

```
    Set-KFSPDatetimeLocal
      [-Item <list-item>]
      [-Field <internal-name-of-field>]
      -DateTimeLocal <datetime-value>
      [-ThrowErrors[:$true]]
```

14.5.88.2 Support

>= *kenaflow* 4.0.12

14.5.88.3 Return

In case of a missing `-Field` it returns the UTC value.

14.5.88.4 Description

This cmdlet can be used to convert a localized DateTime value to UTC. The command assumes that the `-DateTimeLocal` passed is in the time zone set on the SharePoint web regional settings.

14.5.88.5 Parameters

`[-Item <list-item>]`

> You can specify a different list item for which you want to create a rich link. Default is the currently processed item.

`[-Field <internal-name-of-field>]`

> Internal name of a field of the current or specified list item that is updated with the converted DateTime value.

`-DateTimeLocal <datetime-value>`

> Required DateTime value that gets converted to UTC.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.88.6 Example

```
$originalValue = (Get-KFSPDatetimeLocal -Field "SomeDateTimeField")

$newValue = $originalValue.addHours(1)

Set-KFSPDatetimeLocal -Field "SomeDateTimeField" -DateTimeLocal $newValue
$item.Update()

Invoke-PnpQuery
```

14.5.88.7 Related Cmdlets

- *Get-KFSPDatetimeLocal*

14.5.89 Set-KFState

14.5.89.1 Syntax

```
Set-KFState
  [-State] <next-state-key>
  [-ThrowErrors[:$true]]
```

14.5.89.2 Support

>= *kenaflow* 1.0.0

14.5.89.3 Return

Nothing.

14.5.89.4 Description

This cmdlet does only work in SharePoint *state machine* workflows!

It is used to change the workflow state of the currently processed list item.

`-State <next-state-key>` specifies the next state.

The states are configured in `_wfconfig.ps1`, e.g.:

```
#...

        States = @{
                "(empty)" = @{
                        ExecutionOrder = -1;
                        Enabled = $false
                };
                "1.0" = @{
                        Title = "State 1.0";
```

(continues on next page)

(continued from previous page)

```
                    };
                    "2.0" = @{
                            Type="query";
                            Query="{{Title}} == string 'Test' & {{Modified}} >␣
↪datetime [[TODAY]] & {{Editor}} != lookupid [[USERID]]"
                    };
                    "3.0" = @{
                            Title = "Last State";
                            Script = "S_laststate.ps1";
                    };
                    "4.0" = @{
                            Enabled = $false;
                            Title = "Ignore";
                    };
            }

    # ...
```

You can use then `Set-KFState "4.0"` to change the list items workflow state to "4.0".

!! You should not do further actions in the workflow after you executed this cmdlet.

```
    # some lines of workflow script code

    if ($x -eq $true) {
            Set-KFState "1.0"
            Stop-KFScript
    }

    # some more lines of workflow script code.

    Set-KFState "2.0"

    # End of File
```

14.5.89.5 Parameters

`[-State] <next-state-key>`

Specifies the next workflow state.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.89.6 Example

See section "Description" above.

14.5.90 Set-KFStorageData

14.5.90.1 Syntax

```
Set-KFStorageData
  [-Key] <key>
  [-Data] <data>
  [-Binary[:$false]]
  [-AdditionalPassphase <string>]
  [-RelatedToWorkflow[:$true]]
  [-RelatedToItem[:$true]]
  [-ThrowErrors[:$true]]
```

14.5.90.2 Support

>= *kenaflow* 1.0.0

14.5.90.3 Return

Nothing.

14.5.90.4 Description

This cmdlet can be used to write data to the *kenaflow* permanent storage.

Please read the article about *Storage*.

You can serialize objects using the PowerShell serializer when **NOT** specifying `-Binary`.

When using `-Binary` you must specify an byte array (`System.Byte[]`) with `-Data`.

If the requested object exist, it will be overwritten. An exception is not thrown.

By default `-RelatedToWorkflow[:$true]` and `-RelatedToItem[:$true]` are used - alone or in combination - to automatically add prefix(es) to your key.

14.5.90.5 Parameters

`[-Key] <string>`

The key that is used to read the serialized data.

`[-Data] <data>`

This is the data to be serialized and persisted.

`[-Binary:[$false]]`

If NOT specified the cmdlet will store the given object as serialized string. This requires that data for the given `Key` will NOT be read with `-Binary` with `Get-KFStorageData`. If specified the cmdlet requires an byte array (`System.Byte[]`) with parameter `-Data`. The binary data is written to the storage. - This requires that data for the given `Key` will be read with `-Binary` with `Get-KFStorageData`.

`[-AdditionalPassPhrase <passphrase>]`

Workflow developers can specify an additional encryption passphrase. This needs to be specified for retrieving the data with `Get-KFStorageData`!

`[-RelatedToWorkflow[:$true]]`

If you do **not** specify `-RelatedToWorkflow:$false` the workflow id will be used as prefix for your `-Key`. This is the default behaviour.

`[-RelatedToItem[:$true]]`

If you do **not** specify `-RelatedToItem:$false` the current item id will be used as prefix for your `-Key`. This is the default behaviour. The parameter is only available for SharePoint workflows! In SharePoint **site** workflows there is no item context.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.90.6 Example

This writes and reads serialized data to the permanent storage.

```
    Set-KFStorageData -Key "kenaflow" -Data @{"is" = "a workflow engine for␣
↪SharePoint, Email and PowerShell"} -RelatedToItem:$false -RelatedToWorkflow:
↪$false -AdditionalPassphare "cool"
    Write-KFLog (Get-KFStorageData -Key "kenaflow" -RelatedToItem:$false -
↪RelatedToWorkflow:$false -AdditionalPassphare "cool")
    Remove-KFStorageData -Key "kenaflow" -RelatedToItem:$false -RelatedToWorkflow:
↪$false
```

14.5.90.7 Related Cmdlets

- *Get-KFStorageData*
- *Remove-KFStorageData*

14.5.91 Set-KFThrowErrorsPreference

14.5.91.1 Syntax

```
    Set-KFThrowErrorsPreference
        [-ThrowErrorsPreference] <bool>
            [-ThrowErrors[:$true]]
```

14.5.91.2 Support

>= *kenaflow* 2.0.0

14.5.91.3 Return

Nothing.

14.5.91.4 Description

This allows you to set the default behavior for *kenaflow* error handling at the level of the current workflow script.

This will overwrite the workflows default, that is read from `_wfconfig.ps1`.

Please read article *Error Handling*.

14.5.91.5 Parameters

`[-ThrowErrorsPreference] <bool>`

The new error handling preference, `$true` or `$false`.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.91.6 Example

```
Set-KFThrowErrorsPreference $true
```

14.5.91.7 Related Cmdlets

- *Get-KFThrowErrorsPreference*

14.5.92 Start-KFSharePointModule

14.5.92.1 Syntax

```
Start-KFSharePointModule
  [-SharePointVersion] <platform-indicator>
  [-ThrowErrors[:$true]]
```

14.5.92.2 Support

>= *kenaflow* 1.0.0

14.5.92.3 Return

Nothing.

14.5.92.4 Description

kenaflow does not allow parallel installation of the SharePoint PnP PowerShell module.

This is because *kenaflow* itself contains these modules for the SharePoint versions 2013, 2016, 2019 and Online.

To enable you to use SharePoint PnP modules for you own PowerShell scrips on the *kenaflow* server **outside of *kenaflow*** we created this cmdlet to load the module for appropriate SharePoint version from *kenaflow* storage.

Bear this in mind: Because the SharEPoint PnP PowerShell modules are part of *kenaflow* and because you can run multiple instances of *kenaflow* on the same computer it is possible, that each *kenaflow* instance can contain its own version of the PnP PowerShell modules.

14.5.92.5 Parameters

`[-SharePointVersion] <platform-indicator>`

Specifies the SharePoint platform for which you want to load the appropriate SharePoint PnP PowerShell module. Valid values are:

- `SP2013`
- `SP2016`
- `SP2019`
- `SPO`

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.92.6 Example

```
Import-Module "C:\Program Files\kenaflow\kenaflow.runtime.dll"
Start-KFSharePointModule -SharePointVersion sp2013
```

14.5.93 Stop-KFPermissionInheritance

14.5.93.1 Syntax

```
Stop-KFPermissionInheritence
  [-Item <list-item>]
  [-CopyCurrentPermissions[:$false]]
  [-ThrowErrors[:$true]]
```

14.5.93.2 Support

>= *kenaflow* 1.0.0

14.5.93.3 Return

Nothing.

14.5.93.4 Description

With this cmdlet, you stop the inheritance of permissions on a list item.

The list item can be specified with parameter `-Item` if the cmdlet should **not** process the current item of the workflow or - in case of a *site* workflow - there is no "current item".

If it is not specified, the list item for which the workflow is currently being executed is used.

If the permission inheritance is already stopped, nothing is done and nothing is changed.

The cmdlet is only available for SharePoint workflows.

14.5.93.5 Parameters

`[-Item <list-item>]`

The list item to be processed.

`[-CopyCurrentPermissions[:$false]]`

If inheritance is stopped it is possible to copy the permission settings of the parent of the list item - this is a folder or a list. If not specified the permissions of the list item are blank after the cmdlet was executed.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.93.6 Example

```
$myItem = Get-PnPListItem -List "Custom List" -Id 1
Stop-KFPermissionInheritance -Item $myItem
Reset-KFPermissionInheritance -Item $myItem
```

14.5.93.7 Related Cmdlets

- *Add-KFPermission*
- *Get-KFPermission*
- *Reset-KFPermissionInheritance*
- *Set-KFPermission*
- *Update-KFPermission*

14.5.94 Stop-KFScript

14.5.94.1 Syntax

For *PowerShell* workflows:

```
Stop-KFScript
```

For *SharePoint* and *email* workflows:

```
Stop-KFScript
  [-StopWorkflow]
  [-StopWeb]
  [-StopList]
  [-StopState]
  [-Failed]
```

14.5.94.2 Support

>= *kenaflow* 2.0.14

14.5.94.3 Return

Nothing.

14.5.94.4 Description

This cmdlet can be used to force a workflow script to spot immediately at this point of execution.

Without any additional parameter the script stops immediately.

- In case of list or state workflows for SharePoint only the current execution of the script for the current list item will be stopped.
- In case of email workflows only the current execution of the script for the current email will be stopped.

In case of SharePoint workflows and **E-Mail workflows** the parameters `-StopWorkflow` can be used to stop the workflow execution for this time. (The workflow will start again later as configured.)

In case of SharePoint state machine workflows and **SharePoint list workflows** the parameter `-StopList` can be used to stop the execution of the current workflow list. This is only relevant if you have specified more than 1 list to be processed by the workflow. Otherwise it will stop the workflow.

In case of SharePoint state machine workflows and **SharePoint list workflows** the parameter `-StopWeb` can be used to stop the execution of the current web. This is only relevant if you have specified more than 1 web to be processed by the workflow. Otherwise it will stop the workflow.

In case of SharePoint state machine workflows the parameter `-StopState` can be used to stop the execution of the current workflow state.

In case of PowerShell workflows there are no parameters besides `-Failed`. You can only stop the whole workflow by using the cmdlet.

To mark the workflow execution as "failed" the parameter `-Failed` can be used.

14.5.94.5 Parameters

`-StopWorkflow`

Stop the execution of the current workflow. No more items will be executed *in the current workflow run*.

`-StopState`

Stop the execution of the current state. No more items *in this state* will be executed *in the current workflow run*.

`-StopList`

Stop the execution of the current workflow list. No more items *in this list* will be executed *in the current workflow run*.

`-StopWeb`

Stop the execution of the current web. No more items *in this web* will be executed *in the current workflow run*.

`-Failed`

Used to mark the execution as "failed".

14.5.94.6 Examples

```
Stop-KFScript
```

This stops the current item for *SharePoint* workflows and *email* workflows. For *PowerShell* workflows it stops the workflow.

14.5.95 Test-KFConfig

14.5.95.1 Syntax

```
Test-KFConfig
  [-ThrowErrors[:$true]]
```

14.5.95.2 Support

>= *kenaflow* 1.0.0

14.5.95.3 Return

A string value that reports success or error messages.

14.5.95.4 Description

This cmdlet is used to test a workflow configuration file (`_wfconfig.ps1`).

It is part of each `_wfconfig.ps1` in the following line at the end of the file:

```
    if(!$kenaflow){import-module "c:\program files\kenaflow\kenaflow.runtime.dll";
↪Test-KFConfig;exit}
```

When you execute the `_wfconfig.ps1` file itself you get an information wether the config file is valid or not.

14.5.95.5 Parameters

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.96 Test-KFItemLock

14.5.96.1 Syntax

```
Test-KFItemLock
  [-Handle] <handle>
  [-ThrowErrors[:$true]]
```

14.5.96.2 Support

>= *kenaflow* 2.0.3

14.5.96.3 Return

Boolean.

`$true` => `<handle>` is valid. The lock exists and "you" own it.

`$false` => `<handle>` is invalid or does not exist or "you" do not own it.

14.5.96.4 Description

This can be used to test an previously gathered lock handle for a SharePoint list item.

Please read this article for further details: *Locking*.

14.5.96.5 Parameters

`[-Handle] <handle>`

This is a lock handle gathered with *Lock-KFItem*.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.96.6 Example

```
$item = Get-PnPListItem -List "Custom List" -Id 1
$lock = Lock-KFItem -Item $item -MaxWaitInSeconds 10
if( $lock -ne $null ) {
        Test-KFItemLock -Handle $lock
        Unlock
} else {
        Write-KFLog "Cannot get a handle to the SharePoint list item."
}
```

14.5.96.7 Related Cmdlets

- *Lock-KFItem*
- *Unlock-KFItem*

14.5.97 Test-KFProcessLock

14.5.97.1 Syntax

```
Test-KFProcessLock
  [-Handle] <handle>
  [-ThrowErrors[:$true]]
```

14.5.97.2 Support

>= *kenaflow* 2.0.3

14.5.97.3 Return

Boolean.

`$true` => `<handle>` is valid. The lock exists and "you" own it.

`$false` => `<handle>` is invalid or does not exist or "you" do not own it.

14.5.97.4 Description

This cmdlet can be used to test the existence of an custom lock.

Please read this article for further details: *Locking*.

14.5.97.5 Parameters

`[-Handle] <handle>`

This is a lock handle gathered with *`Lock-KFProcess`*.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.97.6 Example

```
$lock = Lock-KFProcess -Key "TestLockKey" -MaxWaitInSeconds 10
if( $lock -ne $null ) {
        Test-KFProcessLock -Handle $lock
        Unlock-KFProcess $lock
} else {
        Write-KFLog "Cannot get a handle for the given key."
}
```

14.5.97.7 Related Cmdlets

- *`Lock-KFProcess`*
- *`Unlock-KFProcess`*

14.5.98 Unlock-KFItem

14.5.98.1 Syntax

```
Unlock-KFItem
  [-Handle] <handle>
  [-ThrowErrors[:$true]]
```

14.5.98.2 Support

>= *kenaflow* 2.0.3

14.5.98.3 Return

Boolean.

`$true` => `<handle>` was a valid lock and it was release!

`$false` => `<handle>` was not a valid lock!

14.5.98.4 Description

You can lock and unlock SharePoint list items from any workflow.

With this cmdlet you release a previously gathered lock.

Please read this article for further details: *Locking*.

14.5.98.5 Parameters

`[-Handle] <handle>`

This is a lock handle gathered with *`Lock-KFItem`*.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.98.6 Example

```
$item = Get-PnPListItem -List "Custom List" -Id 1
$lock = Lock-KFItem -Item $item -MaxWaitInSeconds 10
if( $lock -ne $null ) {
        Test-KFItemLock -Handle $lock
        Unlock
} else {
        Write-KFLog "Cannot get a handle to the SharePoint list item."
}
```

14.5.98.7 Related Cmdlets

- *`Lock-KFItem`*
- *`Test-KFItemLock`*

14.5.99 Unlock-KFProcess

14.5.99.1 Syntax

```
Unlock-KFProcess
  [-Handle] <handle>
  [-ThrowErrors[:$true]]
```

14.5.99.2 Support

>= *kenaflow* 2.0.3

14.5.99.3 Return

Boolean.

`$true` => `<handle>` was a valid lock and it was release!

`$false` => `<handle>` was not a valid lock!

14.5.99.4 Description

This cmdlet can be used to unlock a previously gathered lock to a custom key.

Please read this article for further details: *Locking*.

14.5.99.5 Parameters

`[-Handle] <handle>`

The lock handle.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.99.6 Example

```
$lock = Lock-KFProcess -Key "TestLockKey" -MaxWaitInSeconds 10
if( $lock -ne $null ) {
        Test-KFProcessLock -Handle $lock
        Unlock-KFProcess $lock
} else {
        Write-KFLog "Cannot get a handle for the given key."
}
```

14.5.99.7 Related Cmdlets

- *Lock-KFProcess*
- *Test-KFProcessLock*

14.5.100 Unprotect-KFData

14.5.100.1 Syntax

```
Unprotect-KFData
  [-EncryptedData] <string>
  [-ThrowErrors[:$true]]
```

14.5.100.2 Support

>= *kenaflow* 2.0.19

14.5.100.3 Return

Depending on the data encrypted with `Protect-KFData` it will return:

- a string
- an object
- an array of byte values

Please read the description below.

14.5.100.4 Description

This cmdlet can be used to decrypt data that was encrypted using *Protect-KFData*

The `encryptionPassphrase` from the global configuration is used for decryption.

If the encrypted data contains an object, that it is stored as CliXML serialized string and will be deserialzed after decryption.

14.5.100.5 Parameters

`[-EncryptedData] <string>`

`<string>` contains the encrypted data.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.100.6 Example

```
$encryptedData = Protect-KFData -Object @{Hello="kenaflow"}
Write-KFLogOutput (Unprotect-KFData $encryptedData)
```

14.5.100.7 Related Cmdlets

- *Protect-KFData*

14.5.101 Update-KFExcelFile

14.5.101.1 Syntax

```
Update-KFExcelFile
  -InFile <excel-file>
  -SheetName <string>
  -AsStream
  [-StartAddress <address>]
  [-EndAddress <address>]
  [-EveryCellProcessor <scriptblock>]
  [-Values <list-of-hashtables>]
  [-ThrowErrors[:$true]]

Update-KFExcelFile
  -InFile <excel-file>
  -SheetName <string>
  -AsBase64
  [-StartAddress <address>]
  [-EndAddress <address>]
  [-EveryCellProcessor <scriptblock>]
  [-Values <list-of-hashtables>]
  [-ThrowErrors[:$true]]

Update-KFExcelFile
  -InFile <excel-file>
  -SheetName <string>
  -OutFile <path>
  [-StartAddress <address>]
  [-EndAddress <address>]
  [-EveryCellProcessor <scriptblock>]
  [-Values <list-of-hashtables>]
  [-ThrowErrors[:$true]]
```

14.5.101.2 Support

>= *kenaflow* 4.0.11

14.5.101.3 Return

`System.Collections.Hashtable` object or `$null`

14.5.101.4 Description

This cmdlet can be used to modify Excel file cell **values**. Formats cannot be modified.

The input Excel file is given as

1. as System.IO.Stream
2. as Base64 encoded byte array
3. or as file on disk, relative to the workflow folder.

There are three different output options:

1. as System.IO.MemoryString by using `-AsStream`
2. as Base64 encoded byte array using `-AsBase64`
3. or as file on disk using `-OutFile <path>`

You can only use one output option.

With `-StartAddress` and `-EndAddress` the worksheet region can be specified. Default - if not specified - is "Every cell starting with `A1`", which is identical with `-StartAddress "A1" -EndAddress $null`

With `-EveryCellProcessor` can be used to process every cell value by using a scriptblock.

With `-Values` you can specify a list of hashtable objects with replacement values for specific cells.

If the modified cell is included in `-Values` and `-EveryCellProcessor` is specified too the value in `-Values` used when calling `-EveryCellProcessor` but the current cell value can be accessed with parameter `$cell.CellValue` in the scriptblock.

14.5.101.5 Parameters

`-InFile <template-excel-file>`

An Excel file to be imported.

1. The file can be specified as Base64 encoded string or
2. as System.IO.Sream object or
3. as a full or relative path to an existing file on disk. It's relative to the current workflow folder.

`-SheetName <string>`

The name of the Excel worksheet.

`-AsStream`

The result is a System.IO.MemoryStream object

`-AsBase64`

The result is a Base64 encoded byte array that represents the Excel file.

`-OutFile <path>`

The result is written to a file on disk. The path can be relative to the current workflow folder.

`[-StartAddress <address>]`

This parameter can be used to specify the worksheet region for the data import. If not specified `A1` is used

`<address>` can be

- an Excel style cell addres, such as `A1` or `ZZ55`.
- an Hashtable of this kind: `@{Row=4; Col=2}` or `@{Row="4"; Col="2"}` (The Excel address of the cell would be `B4`.)

`[-EndAddress <address>]`

This parameter can be used to specify the worksheet region for the data import. If not specified the worksheet is processed completely (starting from `-StartAddress`). The format `<address>` is described above.

`[-EveryCellProcessor <scriptblock>]`

This scriptblock can be used to process and manipulate every cell in the specified range of the worksheet.

This is an example of who the script show look like, especially to get the corret parameters. If the script does not return a value the cell will not be modified! An empty string `""` IS a valid result and will replace the cells value.

```
-EveryCellProcessor {
    param($cell, $row, $col, $currentVal)
    if( $row -eq 2 -and $col -eq 3 ) {
        return "Hello User!"
    }
}
```

```
>

`[-Values <list-of-hashtables>]`
> This parameter can be used to specify a list of replacement values, referenced␣
↪by a cell address for each value.
>
> Such a `<list-of-hashtables>` looks like this:
> ```powershell
    -Values @(
       @{Cell = "B2"; Value = "kenaflow"}
       @{Cell = @{Row="3"; Col=3}; Value = "is"}
       @{Cell = "D5"; Value = "cool!"}
    )
```

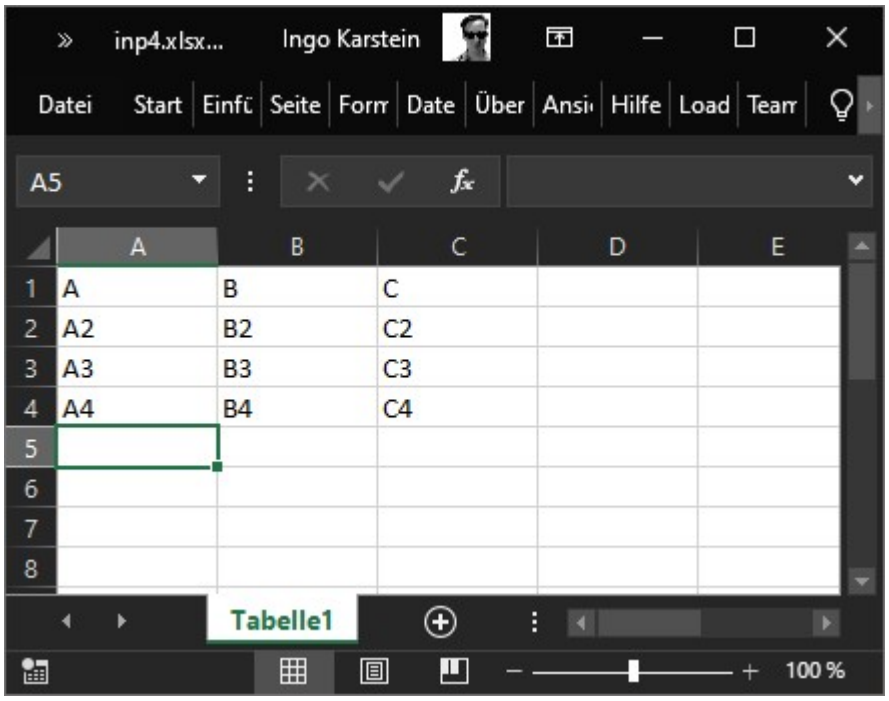
`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

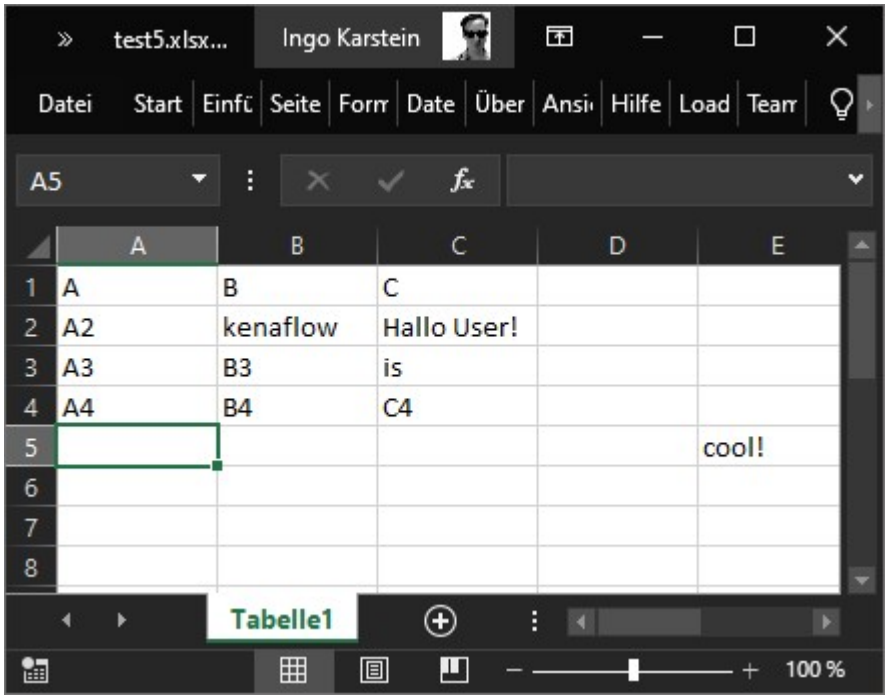
14.5.101.6 Example

```
    Update-KFExcelFile -InFile "$PSScriptRoot\inp4.xlsx" -SheetName "Tabelle1" -
↪StartAddress "A2" `
       -Values @(
            @{Cell = "B2"; Value = "kenaflow"}
            @{Cell = @{Row="3"; Col="3"}; Value = "is"}
            @{Cell = "D5"; Value = "cool!"}
       ) -EveryCellProcessor {
        param($cell, $row, $col, $currentVal)
            if( $row -eq 2 -and $col -eq 3 ) {
                return "Hallo User!"
            }
       } -OutFile "$PSScriptRoot\test5.xlsx"
```

Input file:



Output:



14.5.101.7 Related Cmdlets

- *Invoke-KFTemplate*
- *Invoke-KFFileTemplate*
- *Import-KFExcelFile*
- *New-KFPDF*
- *Import-KFExcelFile*

14.5.102 Update-KFPermission

14.5.102.1 Syntax

```
Update-KFPermission
  [-AddPermissions] <list-of-permissions>
  [-RemovePermissions] <list-of-permissions>
  [-Item <list-item>]
  [-CopyCurrentPermissions[:$false]]
  [-BreakPermissionInheritance[:$true]]
  [-ThrowResolveErrors[:$false]]
  [-ThrowErrors[:$true]]
```

14.5.102.2 Support

>= *kenaflow* 1.0.0

14.5.102.3 Return

Nothing.

14.5.102.4 Description

Use this cmdlet to add and remove permissions to and from a list item. It's a combination of the cmdlets *Add-KFPermission* and *Remove-KFPermission*.

Both parameters `-AddPermissons` and `-RemovePermissions` gets a single permission object or a list of permission objects of the following form.

The permission object is specified as a PowerShell Hashtable:

```
@{
   User = <string-or-object>;
   PermissionsSets = @(<permission-set-name>[, <permission-set-name>[, ...]]);
   Quick = "CRUDMA";
)
```

The `User` property of the permission object can be:

- List of strings or objects
- string, that describe the user
- object as (array of) SharePoint `FieldUserValue`

Such a "string" of the `User` property of the permission object can be:

- a user login name

- an email address
- a SharePoint field of the currently processed or with `-Item` specified list item
- a combination of all of them separated by "," or ";"

Examples:

- `User = @($item["Author"])`
- `User = $item["Author"]`
- `User = @($item["Author"], $item["Editor"])`
- `User = "{{Author}}"`
- `User = "{{Author}},{{Editor}}"`
- `User = @($item["Author"], "{{Editor}},sharepoint\ingo", "ingo@sharepoint.farm")`

The *kenaflow* template engine is used to resolve the given strings. *Templates*

Also SharePoint groups are resolved. E.g. `User = @("Intranet Owners")`

The parameter `-PermissionSets` is a PowerShell list of one or more names of valid / existing SharePoint permission sets, such as "Full Control" or "Design". - Custom permission sets can be named too. But the have to exist!

The parameter `-Quick` is a kind of shortcut to `-PermissionSets`. *kenaflow can* create it's own permission sets as specified in the `_wfconfig.ps1` file. They have to be created using program switch *`--createpermissionsets`*. This permission sets are by default named as:

- kenaflow Create => `C`
- kenaflow Read => `R`
- kenaflow Update => `U`
- kenaflow Delete => `D`
- kenaflow Manage => `M`
- kenaflow Approve => `A`

The first letter(s) **in combination** can be used in parameter `-Quick` to specify these permission sets. - But they have to be configured in `_wfconfig.ps1` and must exist in SharePoint! - For example: `Quick="CUD"` => "Create", "Update", "Delete" - but not "Read".

The base permissions of each of the *kenaflow* defined permission sets can be found here: *Permissions*.

This is an example for a valid permission object - if the *kenaflow* permission sets are configured.

```
@{
    User = @("{{Author}}", $item["Editor"]);
    PermissionSets = @("Read");
    Quick = "CUD"
}
```

Multiple permission objects can be specified as a list:

```
@(
    @{User="sharepoint\ingo"; Quick="CUD"},
    @{User="sharepoint\kenaflow";PermissionSets=@("Full Control")}
)
```

14.5.102.5 Parameters

`[-AddPermissions] <list-of-permissions>`

One or more permission objects as described above to be added to the list item.

`[-RemovePermissions] <list-of-permissions>`

One or more permission objects as described above to be removed from the list item.

`[-Item <list-item>]`

The list item to be processed.

`[-BreakPermissionInheritance[:$true]]`

Permissions can only be added to the list item if it has its own permissions - and not inherited permissions from its parent. If an item with active permission inheritance is given the inheritance will be stopped by default. But if you specify `-BreakPermissionInheritance:$false` the cmdlet will fail and stop if the list item has active permission inheritance.

`[-CopyCurrentPermissions[:$false]]`

If the cmdlet stop permission inheritance it can copy the permissions of the parent or start with a blank set of permissions which is the default.

`[-ThrowResolveErrors[:$false]]`

kenaflow tries to resolve the given `User` property of each permission object. If this fails and `-ThrowResolveErrors` is specified the cmdlet will fail. Otherwise the not existing or not successfully resolved principal is skipped.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.102.6 Examples

```
$newEditor = "sharepoint\user3"

Update-KFPermission -AddPermissions @(
    @{
        User = "user1@sharepoint.farm";
        PermissionSets = @("Full Control");
        Quick="MA";
    },
    @{
        User = "user2@sharepoint.farm";
        PermissionSets = "Read";
        Quick="D";
    }
) -RemovePermissions @(
    @{ User="{{Author}}"; Quick="R" },
    @{ User="{{currentEditor}}"; Quick="R" },
    @{ User=$newEditor; Quick="R" }
)
```

14.5.102.7 Related Cmdlets

- *Add-KFPermission*
- *Get-KFPermission*
- *Reset-KFPermissionInheritance*
- *Set-KFPermission*
- *Stop-KFPermissionInheritance*

14.5.103 Write-KFHistory

14.5.103.1 Syntax

```
Write-KFHistory
  [-Message] <message>
  [-ThrowErrors[:$true]]
```

14.5.103.2 Support

>= *kenaflow* 1.0.0

14.5.103.3 Return

Nothing.

14.5.103.4 Description

Use this cmdlet to write an entry to the Workflow *History List*.

This will fail if the workflow has no configured history list!

14.5.103.5 Parameters

`-Message <message>`

The message to be logged.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.103.6 Examples

```
Write-KFHistory "Hello kenaflow!"
```

14.5.103.7 Related Cmdlets

- *Write-KFLog*
- *Write-KFLogOutput*

14.5.104 Write-KFLog

14.5.104.1 Syntax

```
Write-KFLog
  [-Message] <message>
  [[-LogLevel] <log-level>]
  [-ThrowErrors[:$true]]
```

14.5.104.2 Support

>= *kenaflow* 1.0.0

14.5.104.3 Return

Nothing.

14.5.104.4 Description

This Cmdlet writes the specified `<message>` to the log of *kenaflow*.

14.5.104.5 Parameters

`-Message <message>`

The message to be logged.

`-LogLevel <log-level>`

Specifies the severity of the log entry.

Possible values are:

- Debug
- Verbose
- Normal
- Warning
- Error
- Critical

`Critical` will cause an error message and e-mail to the administrators.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.104.6 Example

```
Write-KFLog "Hello kenaflow!"
```

14.5.104.7 Related Cmdlets

- *Write-KFHistory*
- *Write-KFLogOutput*

14.5.105 Write-KFLogOutput

14.5.105.1 Syntax

```
Write-KFLogOutput
  [-InputObject] <object>
  [-LogOutputFormat <log-output-format>]
  [-LogLevel <log-level>]
  [-ThrowErrors:[$true]]
```

14.5.105.2 Support

>= *kenaflow* 1.0.0

14.5.105.3 Return

Nothing.

14.5.105.4 Description

This Cmdlet writes the given `<object>` in serialized form to the log of *kenaflow*.

By default, the format 'JSON' is used for serialization. Alternatively, 'CliXML' is possible for serialization as PowerShell compatible object serialization XML.

14.5.105.5 Parameters

`-InputObject <object>`

The `<object>` to be logged in serialized form

`-LogOutputFormat <log-output-format>`

The serialization format for the given `<object>`

Possible values are:

- JSON
- CliXML

Default (if not specified) is "JSON"

`-LogLevel <log-level>`

Specifies the severity of the log entry.

Possible values are:

- Debug
- Verbose
- Normal
- Warning
- Error
- Critical

`Critical` will cause an error message and e-mail to the administrators.

`[-ThrowErrors[:$true]]`

This is a default parameter for all *kenaflow* cmdlets. If set to `$true` (default!) the engine will pass exception within the cmdlet to the script for further handling.

14.5.105.6 Example

```
Write-KFLogOutput @{Hello="kenaflow"}
```

14.5.105.7 Related Cmdlets

- *Write-KFHistory*
- *Write-KFLog*

CHAPTER

FIFTEEN

COMMAND LINE REFERENCE

The command line arguments of `kenaflow.exe` are divided into groups.

- *Runtime*
- *Setup and Operation*
- *Workflow Tools*
- *Licensing*
- *Tools*
- *Advanced*
- *Other*
- *Old*

15.1 Runtime

The runtime program switches are used to execute *kenaflow* workflows and other tasks related to the workflow execution.

- *--run*

  Executes the "main mode" that controls all other processes.

- *--exec*

  Executes 1 workflow.

- *--rer*
- *--norer*
- *--norerspo*
- *--norersp2013*
- *--norersp2016*
- *--norersp2019*
- *--norerposh*
- *--rerrunnersp2013*
- *--rerrunnersp2016*
- *--rerrunnersp2019*
- *--rerrunnerspo*
- *--rerrunnerposh*

15.1.1 –run

15.1.1.1 Syntax

```
    kenaflow --run [--norestart] [--onlywf <list-of-workflows>] [--norersp2013] [--
↪norersp2016] [--norersp2019] [--norerspo] [--norerposh] [--norer] `
                    [--noignore] [--notbe] [--currentdirectory <folder>] [--
↪workflowroot <folder>]
```

15.1.1.2 Description

This is the switch to run *kenaflow* in "main mode" to execute all workflows in the workflow root folder.

Please read about *the Details*.

With `--onlywf` you can select workflows from the workflow root foldert that should execute. If this option is used than all workflows ***not*** specified here will be ignored. (Disalbed workflows are also ignored.) Multiple workflows are separated by `*`.

With `--noignore` you can force *kenaflow* to execute a workflow that is disabled (`Enabled = $false` in its `_wfconfig.ps1`).

With `--notbe` you can force *kenaflow* to execute a workflow that would not run in "main mode" (*--run*) also when it next runtime is not come.

!!! `--notbe` and `--noignore` should be used together with `--onlywf` to prevent running ALL workflow.

The following switches starting with `--norer...` disable certain remote event runner processes. Please read these articles for details:

- *--norersp2013*
- *--norersp2016*
- *--norersp2019*
- *--norerspo*
- *--norerposh*

When specifying `--norer` the "global remote event receiver" is not startet.

With `--currentdirectory <folder>` the working directory of the process can be set.

With `--workflowroot <folder>` you set the workflow root folder. This overwrites the Global Configuration setting in *Section "AfterSetup"* for the current execution.

15.1.1.2.1 Examples

This will execute wokflows "WF1" and "WF2" in the workflow root. All other are ignored.

```
    kenaflow --run --onlywf "c:\workflows\wf1*c:\workflows\wf2"
```

15.1.2 –exec

15.1.2.1 Syntax

```
    kenaflow --exec [--workflow <path-or-folder>] [--notbe] [--noignore] [--slot
↪<slot-id>] [--onlywf <list-of-workflows>] [--currentdirectory <folder>] [-plugin
↪<path-to-dll>]
```

15.1.2.2 Description

The switch `--exec` is used to run a kenaflow workflow.

If you run `kenaflow.exe --exec` from within a workflow folder this workflow is executed. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--noignore` you can force *kenaflow* to execute a workflow that is disabled (`Enabled = $false` in its `_wfconfig.ps1`).

With `--notbe` you can force *kenaflow* to execute a workflow that would not run in "main mode" (`--run`) also when it next runtime is not come.

`--slot <slot-id>` is used to assign the workflow to an execution slot. (Please read *this article* for details.)

!!! `--exec` is used by "main mode" to run single workflows in dedicated processes. You will see multiple processes in the Windows Task Manager.

You can have custom Plugins that can offer additional functionality such as custom workflow types. To run a *kenaflow* workflow with a certain plugin you can use this command line switch. Even if plugins are disabled on the *kenaflow* instance (Global Configuration Section **TODO**) the plugin will be executed. See

15.1.3 –rer

15.1.3.1 Syntax

```
    kenaflow --rer [--workflow <path-or-folder>] [--notbe] [--noignore] [--slot
↪<slot-id>] [--onlywf <list-of-workflows>] [--currentdirectory <folder>]
```

15.1.3.2 Description

The switch `--rer` start kenaflow in the "global remote event receiver" mode.

This mode creates a process with an a couple of http endpoints for all types of http based remote events.

One endpoint is registered with SharePoint to receive SharePoint remove events.

Another enpoint can be used start a PowerShell workflow or SharePoint "site workflow" remotely by sending data to them.

Please read article *Remote Event*.

!!! Each kenaflow instance can only have one "global remote event receiver" process!

The process stores the received events in the queue folder as encrypted files. This folder can be configured using key "rerQueueDir" in the *Global Configuration Section "RER"*.

The http endpoints can be secured by using SSL. See command line switch *`--confighttp`*.

15.1.4 –norer

15.1.4.1 Syntax

```
kenaflow.exe --run --norer
```

15.1.4.2 Description

This switch disables "SharePoint Online Workflow Runners" when running `kenaflow.exe` in "run" mode, that executes workflows automatically.

Please read the description of switch *--run*

15.1.4.3 Related switches

- *--run*

15.1.5 –norerspo

15.1.5.1 Syntax

```
kenaflow.exe --run --nonrerspo
```

15.1.5.2 Description

This switch disables "SharePoint Online Workflow Runners" when running `kenaflow.exe` in "run" mode, that executes workflows automatically.

Please read the description of switch *--run*

15.1.5.3 Related switches

- *--run*
- *--norersp2013*
- *--norersp2016*
- *--norersp2019*
- *--norerposh*

15.1.6 –norersp2013

15.1.6.1 Syntax

```
kenaflow.exe --run --nonrersp2013
```

15.1.6.2 Description

This switch disables "SharePoint 2013 Workflow Runners" when running `kenaflow.exe` in "run" mode, that executes workflows automatically.

Please read the description of switch *--run*

15.1.6.3 Related switches

- *--run*
- *--norersp2016*
- *--norersp2019*
- *--norerspo*
- *--norerposh*

15.1.7 –norersp2016

15.1.7.1 Syntax

```
kenaflow.exe --run --nonrersp2016
```

15.1.7.2 Description

This switch disables "SharePoint 2016 Workflow Runners" when running `kenaflow.exe` in "run" mode, that executes workflows automatically.

Please read the description of switch *--run*

15.1.7.3 Related switches

- *--run*
- *--norersp2013*
- *--norersp2019*
- *--norerspo*
- *--norerposh*

15.1.8 –norersp2019

15.1.8.1 Syntax

```
kenaflow.exe --run --nonrersp2019
```

15.1.8.2 Description

This switch disables "SharePoint 2019 Workflow Runners" when running `kenaflow.exe` in "run" mode, that executes workflows automatically.

Please read the description of switch *--run*

15.1.8.3 Related switches

- *--run*
- *--norersp2013*
- *--norersp2016*
- *--norerspo*
- *--norerposh*

15.1.9 –norerposh

15.1.9.1 Syntax

```
kenaflow.exe --run --nonrerposh
```

15.1.9.2 Description

This switch disables "PowerShell Workflow Runners" when running `kenaflow.exe` in "run" mode, that executes workflows automatically.

Please read the description of switch *--run*

15.1.9.3 Related switches

- *--run*
- *--norersp2013*
- *--norersp2016*
- *--norersp2019*
- *--norerspo*

15.1.10 –rerrunnersp2013

15.1.10.1 Syntax

```
kenaflow.exe --rerrunnersp2013
```

15.1.10.2 Description

This switch starts the "remote event runner" mode for SharePoint 2013 workflows.

It is startet by the "main mode". Please read the description of switch *--run*

Please read about *the details* how *kenaflow* is working.

15.1.10.3 Related switches

- *--run*
- *--rerrunnersp2016*
- *--rerrunnersp2019*
- *--rerrunnerspo*
- *--rerrunnerposh*

15.1.11 –rerrunnersp2016

15.1.11.1 Syntax

```
kenaflow.exe --rerrunnersp2016
```

15.1.11.2 Description

This switch starts the "remote event runner" mode for SharePoint 2016 workflows.

It is startet by the "main mode". Please read the description of switch *--run*

Please read about *the details* how *kenaflow* is working.

15.1.11.3 Related switches

- *--run*
- *--rerrunnersp2013*
- *--rerrunnersp2019*
- *--rerrunnerspo*
- *--rerrunnerposh*

15.1.12 –rerrunnersp2019

15.1.12.1 Syntax

```
kenaflow.exe --rerrunnersp2019
```

15.1.12.2 Description

This switch starts the "remote event runner" mode for SharePoint 2019 workflows.

It is startet by the "main mode". Please read the description of switch *--run*

Please read about *the details* how *kenaflow* is working.

15.1.12.3 Related switches

- *--run*
- *--rerrunnersp2013*
- *--rerrunnersp2016*
- *--rerrunnerspo*
- *--rerrunnerposh*

15.1.13 –rerrunnerspo

15.1.13.1 Syntax

```
kenaflow.exe --rerrunnerspo
```

15.1.13.2 Description

This switch starts the "remote event runner" mode for SharePoint Online workflows.

It is startet by the "main mode". Please read the description of switch *--run*

Please read about *the details* how *kenaflow* is working.

15.1.13.3 Related switches

- *--run*
- *--rerrunnersp2013*
- *--rerrunnersp2016*
- *--rerrunnersp2019*
- *--rerrunnerposh*

15.1.14 –rerrunnerposh

15.1.14.1 Syntax

```
kenaflow.exe --rerrunnerposh
```

15.1.14.2 Description

This switch starts the "remote event runner" mode for PowerShell workflows.

It is startet by the "main mode". Please read the description of switch *--run*

Please read about *the details* how *kenaflow* is working.

15.1.14.3 Related switches

- *--run*
- *--rerrunnersp2013*
- *--rerrunnersp2016*
- *--rerrunnersp2019*
- *--rerrunnerspo*

15.2 Setup and Operation

- *--editconfig*
- *--securing*
- *--installschedtask*
- *--createperformancecounter*
- *--removeperformancecounter*
- *--testperformancecounter*
- *--decryptconfig*
- *--configthttp*
- *--sendtestmail*
- *--updatenetsecurity*

15.2.1 –enable

15.2.1.1 Syntax

```
kenaflow.exe --enable
```

15.2.1.2 Description

This command is used to re-enable kenaflow if it was disabled before using *--disable*

! This command line switch must be run "as administrator" (high priviledges).

15.2.2 –disable

15.2.2.1 Syntax

```
kenaflow.exe --disable "<time-span>"
```

15.2.2.2 Description

This command can be used to disable *kenaflow* for the time specified in `<time-span>`.

"Disable" means that *kenaflow* will not work and exit immediately.

`<time-span>` requires a setting in a format that is understood by static method `System.TimeSpan.Parse`.

Examples:

- `1.0:0:0` => 1 Day
- `1:0` => 1 Hour
- `0:10` => 10 Minutes

This command creates a file named `.disabled` in the *kenaflow* program folder. The file contains the date and time at which *kenaflow* will be enabled again.

_kenaflow can be enabled with *`--enable`*.

! This command line switch must be run "as administrator" (high priviledges).

15.2.3 –editconfig

15.2.3.1 Syntax

```
kenaflow.exe --editconfig
```

15.2.3.2 Description

This command opens the global configuration file in an editor.

The global configuration file is called `kenaflow.conf` and is located in the same directory as `kenaflow.exe`.

By default, the configuration file is encrypted. In order for it to be edited, it must be temporarily stored on the hard disk. After closing the editor (not after saving the temporary file) the temporary file is read, encrypted and deleted.

The instance of *kenaflow* active at this time, including all subprocesses, recognizes the change to the global configuration file and terminates and restarts the processes. This ensures that the changes to the global configuration file are taken into account.

! This command line switch must be run "as administrator" (high priviledges).

15.2.4 –changeconfigsetting

15.2.4.1 Syntax

```
kenaflow.exe --changeconfigsetting --name <name> --value <value>
```

15.2.4.2 Description

This command changes settings in the global configuration file.

`<name>` is the name of the config setting to be changed.

`<value>` is the new value for the setting.

If the setting is already known to *kenaflow* - already included in `kenaflow.conf` - the `<value>` is parsed to match the data type of the setting. E.g. `System.Boolean`. If the `<value>` cannot be parsed an error is shown. No changes are made then.

If the setting is unknown a new entry is created in `kenaflow.conf` in section `new`. The data type of such a new setting is always `System.String`.

After a change was made the active instance of *kenaflow*, including all subprocesses, recognizes the change to the global configuration file and terminates and restarts the processes. This ensures that the changes to the global configuration file are taken into account.

! This command line switch must be run "as administrator" (high priviledges).

15.2.5 –securing

15.2.5.1 Syntax

```
kenaflow.exe --securing --account <account-name>
```

15.2.5.2 Description

After setup or during administration of *kenaflow* this switch is used to set permissions on file system folders in order to keep *kenaflow* running.

The given `<account-name>` must be a valid user object that can be resolved by *kenaflow*.

It sets read and write permissions on certain folders.

!! You must run this switch for ***every user account*** that should be able to execute workflows - even as workflow developer - as long as this accounts are not "local admin" on the *kenaflow* server.

If you use *--confighttp* to register an SSL certificate for remote event endpoints access to the private key is necessary for the automatic execution account of *kenaflow*. This is granted by this command IF the SSL certificate was registered previously (and it's thumbprint was written to the global configuration, setting 'sslCert-Thumbprint').

15.2.5.3 Example

```
kenaflow.exe --securing --account "mydomain\service.kenaflow"
```

15.2.6 –installschedtask

15.2.6.1 Syntax

```
kenaflow.exe --installschedtask --account <account-name> --password <password>
```

15.2.6.2 Description

kenaflow itself is not a windows service. It's just a command line "tool".

We suggest to register it as "Windows Scheduled Task". - Run `taskschd.msc` to open the task scheduler.

To register *kenaflow* as such "Windows scheduled task" simply run:

```
kenaflow.exe --installschedtask --account "MYDOMAIN\svc.kenaflow" --password
↪"kr@ftwerk"
```

The account is used to execute *kenaflow*. The password is required to register the task to run in the background without manual logon.

!! By default this account will also be used to access SharePoint if a SharePoint related workflow has no own configuration for credentials!

15.2.7 –createperformancecounter

15.2.7.1 Syntax

```
kenaflow.exe --createperformancecounter
```

15.2.7.2 Description

kenaflow can provide execution information via Windows performance counters.

With this program switch the performance counters gets registered with windows.

15.2.8 –removeperformancecounter

15.2.8.1 Syntax

```
kenaflow.exe --removeperformancecounter <version>
```

15.2.8.2 Description

kenaflow can provide execution information via Windows performance counters.

With this program switch the performance counters are removed from windows for a specific version of *kenaflow*.

15.2.8.3 Example

```
kenaflow.exe --removeperformancecounter "4.0.1"
```

15.2.9 –testperformancecounter

15.2.9.1 Syntax

```
kenaflow.exe --testperformancecounter
```

15.2.9.2 Description

kenaflow can provide execution information via Windows performance counters.

With this program switch the performance counters are tested. Sample data is send to each of them.

15.2.10 –decryptconfig

15.2.10.1 Syntax

```
kenaflow.exe --decryptconfig
```

15.2.10.2 Description

This command decrypts the global configuration file.

The global configuration file is called `kenaflow.conf` and is located in the same directory as `kenaflow.exe`.

This switch is not available in FREE edition and DEVELOPER edition of *kenaflow*. There the global configuration is not encrypted on disk.

15.2.11 –confighttp

15.2.11.1 Syntax

```
kenaflow.exe --confighttp --thumbprint <thumbprint-as-hex-string> -ipport <ip-
↪address:port-as-int> [--hostnameport <host-name:port-as-int>] [--certstore <cert-
↪store>]
```

15.2.11.2 Description

This command line switch is used to register an SSL certificate for the "Remote Event Receiver" TCP HTTP endpoint.

The certificate is specified using its thumbprint.

!! If you copy the thumbprint out of the certificate property window please make sure that no hidden characters are copied! This is a known problem! First copy the thumbprint to `notepad.exe` and select all characters from the first HEX digit to the last.

!! Update to the last note: We have implemented a workaround: all characters other than `0..9` and `a..f` and `A..F` are removed from the thumbprint. Now it should work!

If `--certstore <cert-store>` is not specified the SSL certificate is taken from the "My" ("Personal") store of the local machine. `WebHosting` is also a common certificate store for SSL certificates.

You need to enable the *kenaflow* execution account to access the private key. Please use command line switch *`--securing`* to grant access for users to the certificate private keys.

!! When changing the Remote Event handler settings in the global config you maybe need to run `kenaflow.exe --updatenetsecurity` to grant access to the new URL. !! Execute it for the *kenaflow* service account to make sure that the account has at least read permissions on the private key of the certificate! This is a known problem!

With `--ipport <ip-address:port-as-int>` you can bind the SSL certificate to an IP/port endpoint.

With `--hostnameport <hostname:port-as-int>` you can bind the SSL certificate to an host name/port endpoint.

You cannot use `--ipport` and `--hostnameport` together. But you use them separately to bind the certificate to multiple endpoints.

15.2.11.3 Example

The following example shows how to bind the certificate to port 443 of IP address 172.16.1.1. No host name is specified.

```
    kenaflow.exe --configthttp --thumbprint "AA-BB-CC-DD-EE-FF-00-11-22-33-44-55-66
↪" --ipport "172.16.1.1:443"
```

The following example shows how to bind the certificate to port 443 of all IP addresses (`0.0.0.0`) with host name "server.kenaflow.com"

```
    kenaflow.exe --configthttp --thumbprint "AABBCCDDEEFF00112233445566" --
↪hostnameport "server.kenaflow.com:443" --certstore "WebHosting"
```

15.2.12 –sendtestmail

15.2.12.1 Syntax

```
    kenaflow.exe --sendtestmail <single-email-address>
```

15.2.12.2 Description

This can be used to test your SMTP settings!

It will send an email with subject

```
kenaflow TEST MAIL
```

and body

```
This is a testmail you just requested.
```

15.2.13 –updatenetsecurity

15.2.13.1 Syntax

>= *kenaflow* 3.0.15

```
kenaflow.exe --updatenetsecurity  --account <account> [--cleanup]
```

15.2.13.2 Description

When changing the Remote Handler URL (ip, port, . . .) you need to update Windows security settings for accessing these http endpoints.

The Windows command "netsh" is used therefore.

With the program mode `--updatenetsecurity` you can add users (one by one!!) with `--account` to the access control list of the http endpoint.

If you want to cleanup the endpoints access control list you can use `--cleanup`.

15.2.13.3 Examples

```
   #run this when you changed RER settings (host, port) in the global␣
↪configuration.
   kenaflow.exe --updatenetsecurity
```

15.2.14 –reset

15.2.14.1 Syntax

>= *kenaflow* 4.0.18

```
kenaflow.exe --reset
```

15.2.14.2 Description

This program mode can be used at an elevated commandline ("Run as Administrator") to stop the current kenaflow processes.

It creates a file called `.reset` in the program folder.

Only a run of *kenaflow* in automatic mode `--run` will remove the file and allow *kenaflow* to run normally. All other starts of *kenaflow* processes will fail.

The file `.reset` can be removed manually!

This program mode is used to stop *kenaflow* and let it re-start automatically throught the Windows task scheduler.

15.2.14.3 Examples

```
kenaflow.exe --reset
```

15.3 Workflow Management

The following command line switches are used to work with workflows

15.3.1 Remote Events

- *--registerrer*
- *--unregisterrer*

15.3.1.1 Management

- *--new*
- *--setusername*
- *--appid*
- *--removecomments*
- *--backupconfiglist*
- *--copy*

15.3.1.2 Initialization

- *--createpermissionsets*
- *--createconfiglist*
- *--createstatelist*
- *--createerrorlist*
- *--createdatalist*
- *--createhistorylist*
- *--showstatelist*
- *--showerrorlist*
- *--showdatalist*

- *--showhistorylist*
- *--hidestatelist*
- *--hideerrorlist*
- *--hidedatalist*
- *--hidehistorylist*
- *--showconfiglist*
- *--hideconfiglist*
- *--addstatefield*

15.3.2 –registerrer

15.3.2.1 Syntax

```
    kenaflow.exe --registerrer [--workflow <path-or-folder>] [--currentdirectory
↪<folder>]  [--web <web-url>]
```

```
    kenaflow.exe --registerrer --all [--workflowroot <folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.2.2 Description

The **first syntax** is used to register remote events with SharePoint for a dedicated workflow.

You can execute `kenaflow.exe --registerrer` from within the workflow folder or you can specify a workflow folder using `--workflow <folder>` or you can specify a workflow config file (`_wfconfig.ps1`) using `--workflow <path>`.

The **second syntax** is used to register remote events for all SharePoint workflows in the workflow root folder that are ENABLED (`Enable = $true`).

Therefore use switch `--all` in addition to `--registerrer`.

With `--workflowroot <folder>` you set the workflow root folder. This overwrites the global configuration setting in *Section "AfterSetup"* for the current execution.

For both options:

With `--currentdirectory <folder>` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed.

15.3.3 –unregisterrer

15.3.3.1 Syntax

```
    kenaflow.exe --unregisterrer [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

```
    kenaflow.exe --unregisterrer --all [--workflowroot <folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.3.2 Description

The **first syntax** is used to unregister remote events with SharePoint for a dedicated workflow.

You can execute `kenaflow.exe --unregisterrer` from within the workflow folder or you can specify a workflow folder using `--workflow <folder>` or you can specify a workflow config file (`_wfconfig.ps1`) using `--workflow <path>`.

The **second syntax** is used to unregister remote events for all SharePoint workflows in the workflow root folder that are ENABLED (`Enabled = $true`).

Therefore use switch `--all` in addition to `--unregisterrer`.

With `--workflowroot <folder>` you set the workflow root folder. This overwrites the global configuration setting in *Section "AfterSetup"* for the current execution.

For both options:

With `--currentdirectory <folder>` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed.

15.3.4 –new

15.3.4.1 Syntax

```
kenaflow.exe --new --wftype <type> [--currentdirectory <folder>]
```

15.3.4.2 Description

With this switch you can create a new workflow in an empty directory.

`<type>` can be:

- statemachine
- list
- site
- email
- posh

During creation an ID is generated for the new workflow.

You need to configure the `_wfconfig.ps1` manually!

For SharePoint related workflows for example you need to configure the SharePoint platform and the web and list.

15.3.5 –setusername

15.3.5.1 Syntax

```
kenaflow.exe --setusername <username> --setpassword <password> [--workflow
↪<path-or-folder>] [--currentdirectory <folder>]
```

15.3.5.2 Description

This can be used to specify credentials for SharePoint access that are stored in encrypted fashion in `_wfconfig.ps1`.

It replaces the properties `spUser` and `spPassword` in `_wfconfig.ps1`.

15.3.6 –appid

15.3.6.1 Syntax

```
    kenaflow.exe --appid <app-id> --appsecret <app-secret> [--workflow <path-or-
↪folder>] [--currentdirectory <folder>]
```

15.3.6.2 Description

15.3.6.3 Syntax

```
    kenaflow.exe --setusername <username> --setpassword <password> [--workflow
↪<path-or-folder>] [--currentdirectory <folder>]
```

15.3.6.4 Description

This can be used to specify app credentials for SharePoint (Online) access that are stored in encrypted fashion in `_wfconfig.ps1`.

It replaces the properties `spoAppId` and `spoAppSecret` in `_wfconfig.ps1`.

15.3.7 –removecomments

15.3.7.1 Syntax

```
    kenaflow.exe --removecomments [--workflow <workflow-folder>]
```

15.3.7.2 Description

When you create a new workflow using *`kenaflow.exe --new --wftype <type>`* than you get a workflow with a `_wfconfig.ps1` with lots of comments and descriptions of properties.

Example:

```
    @{
        #* A workflow will only work with the following version of
        #*   kenaflow. After you tested your workflow with a new
        #*   version of kenaflow you can change the following setting.
        Version = "4.0";

        #* set it to $false to ENABLE the workflow. ($true DISABLES it)
        Enabled = $false;

        #* Type = SHAREPOINT, EMAIL, PowerShell
        Type = "POWERSHELL";

        #* set it to a meaningful name
```

(continues on next page)

(continued from previous page)

```
        Name = "<<wfname>>";

        #* a unique (!) id (type 'System.Guid') for this workflow.
        #*   must be unique in YOUR environment.  Use
        #*   "[guid]::newGuid().tostring('d')" to create a GUID in
        #*   Powershell.
        Id = "<<wfid>>";

    #... more properties
    }
```

When execution the command `kenaflow.exe --removecomments` from within the workflow folder or with additional switch `--workflow <folder>` where `<folder>` is the path to the workflow folder or the workflows `_wfconfig.ps1` than all lines starting with `#*` are removed. The result `_wfconfig.ps1` looks like this:

```
    @{
        Version = "4.0";

        Ignore = $true;

        Type = "POWERSHELL";

        Name = "<<wfname>>";

        Id = "<<wfid>>";

    #... more properties
    }
```

15.3.7.3 Related Switches

- *--new*

15.3.8 –backupconfiglist

15.3.8.1 Syntax

```
    kenaflow.exe --backupconfiglist <output-file-path> [--workflow <path-or-folder>
↪] [--currentdirectory <folder>] [--web <web-url>]
```

15.3.8.2 Description

If the current workflow has a config list than you are able to export its current values to a PowerShell file. This file you can (manually) copy into the `_wfconfig.ps1` file to keep it up-to-date.

The config list is intended to be used by "process owners" - not only workflow developers. Therefore "process owner" (that for example could change an email templatE) are not able to write the change to the `_wfconfig.ps1` file. With this command the workflow developer is able to simply store a copy of the current values of the config list in the workflow folder.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed.

15.3.8.3 Example

```
    kenaflow.exe --backupconfiglist "current-values-$([DateTime]::Now.ToString(
↪"yyyy-MM-dd-HH-mm-ss")).ps1" --currentdirectory "C:\Workflows\WF1"
```

This will export the current values of the config list of workflow "WF1" to a file with a name starting with "current-values-"...

15.3.9 –copy

15.3.9.1 Syntax

```
    kenaflow.exe --copy  --dest <folder> [--workflow <path-or-folder>] [--
↪currentdirectory <folder>]
```

15.3.9.2 Description

This switch allows you to copy the current workflow to a new folder.

If you run `kenaflow.exe --copy` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

Use `--dest <folder>` to specify the destination.

!! Duing copy the new workflow will be disabled (`Enabled = $false`) and it will get a new ID.

15.3.10 –createpermissionsets

15.3.10.1 Syntax

```
    kenaflow.exe --createpermissionsets [--forcesitecollection] [--workflow <path-
↪or-folder>] [--currentdirectory <folder>] [--web <web-url>]
```

15.3.10.2 Description

With this switch you can use *kenaflow* to create SharePoint permission sets in the SharePoint site that is connected to the workflow.

If you run `kenaflow.exe --createpermissionsets` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

!! If your workflow is connected to a sub site of a site collection than it is not possible with the SharePoint "Client Side Object Model" to create the permission sets in the sub site. You have to create them at site collection level. To do this use the additional switch `--forcesitecollection`.

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed.

This switch can be combined in a single execution of `kenaflow.exe` all switches in section "Related Switches" below.

15.3.10.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createstatelist*
- *--createerrorlist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--addstatefield*

15.3.11 –createconfiglist

15.3.11.1 Syntax

```
    kenaflow.exe --createconfiglist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.11.2 Description

With this switch you can create the **config list** for a workflow as long as the workflow has configured a config list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --createconfiglist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

When running this switch the values from the `_wfconfig.ps1` file are written into the config list. If this list already exists it will be updated: NEW values from `_wfconfig.ps1` are written to the config list. But existing keys are never touched!

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.11.3 Related Switches

- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.12 –createstatelist

15.3.12.1 Syntax

```
    kenaflow.exe --createstatelist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.12.2 Description

With this switch you can create the **state list** for a state machine workflow. The lists name is configured in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --createstatelist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.12.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.13 –createerrorlist

15.3.13.1 Syntax

```
    kenaflow.exe --createerrorlist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.13.2 Description

With this switch you can create the **error list** for a workflow as long as the workflow has configured an error list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --createerrorlist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.13.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.14 –createdatalist

15.3.14.1 Syntax

```
    kenaflow.exe --createdatalist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.14.2 Description

With this switch you can create the **data list** for a workflow as long as the workflow has configured a data list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --createdatalist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.14.3 Related Switches

- *--createconfiglist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.15 –createhistorylist

15.3.15.1 Syntax

```
    kenaflow.exe --createhistorylist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.15.2 Description

With this switch you can create the **history list** for a workflow as long as the workflow has configured a history list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --createhistorylist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.15.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.16 –showstatelist

15.3.16.1 Syntax

```
    kenaflow.exe --showstatelist [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.16.2 Description

With this switch you can make the **state list** visible - as long as the workflow has configured a state list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --showstatelist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.16.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.17 –showerrorlist

15.3.17.1 Syntax

```
    kenaflow.exe --showerrorlist [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.17.2 Description

With this switch you can make the **error list** visible - as long as the workflow has configured a error list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --showerrorlist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.17.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.18 –showdatalist

15.3.18.1 Syntax

```
    kenaflow.exe --showdatalist [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.18.2 Description

With this switch you can make the **data list** visible - as long as the workflow has configured a data list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --showdatalist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.18.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.19 –showhistorylist

15.3.19.1 Syntax

```
    kenaflow.exe --showhistorylist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.19.2 Description

With this switch you can make the **history list** visible - as long as the workflow has configured a history list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --showhistorylist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.19.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.20 –hidestatelist

15.3.20.1 Syntax

```
    kenaflow.exe --hidestatelist [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.20.2 Description

With this switch you can hide the **state list** - as long as the workflow has configured a state list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --hidestatelist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.20.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--createpermissionsets*
- *--addstatefield*

15.3.21 –hideerrorlist

15.3.21.1 Syntax

```
    kenaflow.exe --hideerrorlist [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.21.2 Description

With this switch you can hide the **error list** - as long as the workflow has configured a error list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --hideerrorlist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.21.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.22 –hidedatalist

15.3.22.1 Syntax

```
    kenaflow.exe --hidedatalist [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.22.2 Description

With this switch you can hide the **data list** - as long as the workflow has configured a data list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --hidedatalist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.22.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.23 –hidehistorylist

15.3.23.1 Syntax

```
    kenaflow.exe --hidehistorylist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.23.2 Description

With this switch you can hide the **history list** - as long as the workflow has configured a history list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --hidehistorylist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.23.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.24 –showconfiglist

15.3.24.1 Syntax

```
    kenaflow.exe --showconfigist [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.24.2 Description

With this switch you can make the **config list** visible - as long as the workflow has configured a config list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --showconfiglist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.24.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.25 –hideconfiglist

15.3.25.1 Syntax

```
    kenaflow.exe --hideconfiglist [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.3.25.2 Description

With this switch you can hide the **config list** - as long as the workflow has configured a config list by name in the `_wfconfig.ps1` file.

If you run `kenaflow.exe --hideconfiglist` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.25.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createerrorlist*
- *--createstatelist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*
- *--addstatefield*

15.3.26 –addstatefield

15.3.26.1 Syntax

```
    kenaflow.exe --addstatefield [--workflow <path-or-folder>] [--currentdirectory
↪<folder>] [--web <web-url>]
```

15.3.26.2 Description

With this switch you can add the state field of a state machine workflow to the SharePoint list connected to the workflow.

If you run `kenaflow.exe --addstatefield` from within a workflow folder you do not need to specify `--workflow <path-or-folder>`. From any other working directory you need to specify `--workflow <path-or-folder>` with eigther a `path` to a `_wfconfig.ps1` file or a `folder` to a workflow folder (that contains a `_wfconfig.ps1` file).

With `--currentdirectory` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

This switch can be combined in a single execution of `kenaflow.exe` all switches in section “Related Switches” below.

15.3.26.3 Related Switches

- *--createconfiglist*
- *--createdatalist*
- *--createhistorylist*
- *--createstatelist*
- *--createerrorlist*
- *--showconfiglist*
- *--showdatalist*
- *--showhistorylist*
- *--showerrorlist*
- *--showstatelist*
- *--hideconfiglist*
- *--hidedatalist*
- *--hidehistorylist*
- *--hideerrorlist*
- *--hidestatelist*
- *--createpermissionsets*

15.3.27 –encryptcustomsetting

```
## empty
```

15.4 Licensing

- *--getactivationcode*
- *--serialcode*
- *--showeula*
- *--iaccepteula*
- *--activate*
- *--deactivate*

15.4.1 –getactivationcode

15.4.1.1 Syntax

```
    kenaflow.exe --getactivationcode
```

15.4.1.2 Description

This command displays the activation code of the computer.

The activation code is identical for all *kenaflow* instances on the computer.

Changes to the (virtual) hardware of the computer can change the activation code, which means that an activated *kenaflow* instance will no longer run. Deactivation must be performed in this case.

15.4.2 –serialcode

15.4.2.1 Syntax

```
kenaflow.exe --serialcode <serial-code-as-string>
```

15.4.2.2 Description

This command registers the serial code for the *kenaflow* instance. The command must be executed separately for each *kenaflow* instance.

15.4.2.3 Example

```
kenaflow.exe --serialcode "1111-2222-3333-4444-5555-6666-7777-8888-9999-AAAA"
```

15.4.3 –showeula

15.4.3.1 Syntax

```
kenaflow.exe --showeula
```

15.4.3.2 Description

This switch will write the "End User License Agreement" to the console window and to the *kenaflow* log.

!! You need to run this to be able to activate *kenaflow*!

15.4.4 –iaccepteula

15.4.4.1 Syntax

```
kenaflow.exe --iaccepteula
```

15.4.4.2 Description

With this switch you accept the "End User License Agreement" of *kenaflow*.

!! You need to run this to be able to activate *kenaflow*!

15.4.5 –activate

15.4.5.1 Syntax

```
kenaflow.exe --activate <email-address>

kenaflow.exe --activate <path-to-license-file>
```

15.4.5.2 Description

This switch will activate *kenaflow*

You can use the **first** syntax if your *kenaflow* server is able to connect to our registration server over internet.

!! Maybe you need to configure file `kenaflow.exe.config` to specify you internet proxy!

You can use the **second** syntax if you have active *kenaflow* "offline" using a browser on another computer. Than you specify the licence file here. You download the license file during "offline" activation.

15.4.6 –deactivate

15.4.6.1 Syntax

```
kenaflow.exe --deactivate
```

15.4.6.2 Description

This command disables all *kenaflow* instances running on a computer under an account.

If your machine is able to connect to our licensing server than your licenses is free for use on another machine.

If you have no internet connection than you need to deactivate *kenaflow* "offline" by using a browser.

15.5 Tools

- *--simpriocalc*
- *--getembeddedimage*
- *--vscode*
- *--cleanuperrorlist*

15.5.1 –simpriocalc

15.5.1.1 Syntax

```
kenaflow.exe --simpriocalc --newtemplate --outfile <path>

kenaflow.exe --simpriocalc --infile <path>
```

15.5.1.2 Description

This is an advanced topic. It is not finished yet.

It can be used to simulate "slots" and "workflow priority" to see how wokflows with given settings (from a control file) are executed: how often and in which order.

The template looks like this:

```
<kenaflorpriocalcsim>
  <runner>3</runner>
  <clients>
    <client prio=""0"" tbe=""10""/>
    <client prio=""0"" tbe=""10""/>
    <client prio=""0"" tbe=""10""/>
    <client prio=""0"" tbe=""10""/>
    <client prio=""0"" tbe=""10""/>
    <client prio=""0"" tbe=""10""/>
  </clients>
</kenaflorpriocalcsim>
```

You can test it, if you want. :-)

Just copy the XML above to a XML file "template.xml" and run:

```
kenaflow.exe --simpriocalc --infile "template.xml"
```

15.5.2 –getembeddedimage

15.5.2.1 Syntax

```
kenaflow --getembeddedimage <input-file-name> --outfile <output-file-name>
```

15.5.2.2 Description

This tool embedded to kenaflow can be used to create embedded image for HTML, e.g. for use in email templates.

The `<input-file>` is a PNG or JPG file with one of these file extensions. Other extensions (including .JPEG, . . .) are currently not supported.

The output file will contain a Base64 encoded image string.

15.5.2.3 Examples

```
kenaflow --getembeddedimage image.jpg --outfile image.txt
```

15.5.3 –vscode

15.5.3.1 Syntax

```
kenaflow.exe --vscode
```

15.5.3.2 Description

This starts Visual Studio Code that is embedded in the *kenaflow* program folder.

15.5.3.3 Example

```
kenaflow --vscode
```

15.5.4 –ulsviewer

15.5.4.1 Syntax

```
kenaflow.exe --ulsviewer
```

15.5.4.2 Description

This will start the ULSViewer and immediately load the live logs of _kenaflow. You do not need to choose the log folder manually. The log file location is taken from the global configuration.

15.5.5 –cleanuperrorlist

15.5.5.1 Syntax

```
kenaflow.exe --cleanuperrorlist <days> [--workflow <path-or-folder>] [--
↪currentdirectory <folder>] [--web <web-url>]

kenaflow.exe --cleanuperrorlist <days> --all [--workflowroot <folder>] [--
↪currentdirectory <folder>] [--web <web-url>]
```

15.5.5.2 Description

The **first syntax** is used clean up the error list of one certain workflow. You specify with ` ´ (an integer greater or equal 0) how old must error entries be to be deleted.

You can execute `kenaflow.exe --cleanuperrorlist` from within the workflow folder or you can specify a workflow folder using `--workflow <folder>` or you can specify a workflow config file (`_wfconfig.ps1`) using `--workflow <path>`.

The **second syntax** is used clean up the error lists of **all enabled** workflows. You specify with `<days>´ (an integer greater or equal 0) how old must error entries be to be deleted. Therefore use switch –allin addition to–cleanuperrorlist`.

With `--workflowroot <folder>` you set the workflow root folder. This overwrites the Global Configuration setting in *Section "AfterSetup"* for the current execution.

For both options:

With `--currentdirectory <folder>` the working directory of the process can be set.

In case of SharePoint workflows that are connected to multiple webs you can specify `--web <web-url>` to specify the web that only should be processed. Otherwise all connected webs are processed!

15.6 Advanced

```
## empty
```

15.7 Other

- *--help*
- *--testmain*

15.7.1 –help

15.7.1.1 Syntax

```
kenaflow.exe --help
```

15.7.1.2 Description

This will show all available switches in the console.

15.7.2 –testmain

15.7.2.1 Syntax

```
kenaflow.exe --testmain
```

15.7.2.2 Description

This is only for tests. It will run several "test client" processes of *kenaflow*, started as `kenaflow.exe --testclient`.

The count of test clients is configured in the Global Configuration.

15.8 Old

```
## empty
```

15.8.1 Work Modes

15.8.1.1 `--run`

This mode executes *kenaflow* in the actual execution mode.

kenaflow launches several subprocesses, that is, other *kenaflow*.exe instances that perform various tasks.

Multiple Tasks are possible:

- Receive remote events from SharePoint
- Processing of remote events from SharePoint 2013
- Processing of remote events from SharePoint 2016
- Processing of remote events from SharePoint 2019
- Processing of remote events from SharePoint Online
- Workflow processing

In the default setting, 1 process for receiving remote events is started, plus one process each for processing remote events for SharePoint 2013, 2016 and Online. There are also 5 processes for workflow processing.

All workflows that *kenaflow* finds in the workflow root directory - as specified in the global configuration file - are queued and processed one after the other. A maximum of 5 in parallel (in the standard setting).

Each individual workflow can only be executed once at a time.

Each SharePoint list item can only be processed by 1 workflow at a time.

For example, the remote event processing processes might want to access list items that are currently being processed. The processing is then postponed by remote event and tried again later.

The number of processing processes can be changed via the global *kenaflow* configuration. See section "multiprocessing".

Together with `--run`, the `--workflowroot` switch can be used to ignore the globally configured workflow root directory and use the specified one instead.

If the global configuration changes during execution, all subprocesses and the main process are terminated and restarted.

If a workflow configuration changes, the subprocess for this workflow is restarted.

15.8.1.2 --workflowroot

will be edited soon

15.8.1.3 --vscode

This starts Visual Studio code that is integrated into the *kenaflow* program directory.

The data and user directories of **Visual Studio Code** are adapted so that all data (including extensions) is stored in the context of the current *kenaflow* instance.

15.8.1.4 --onlywf

The `--onlywf=<full-path>[%<full-path>[%<full-path>...] option` can be used to allow only special workflows in subprocesses. The paths to workflow configuration files must be specified. Several are separated by `%` if necessary.

Execution delay means that for each workflow or additionally for each workflow step (for state machine workflows) you can specify how much time should elapse between two executions.

15.8.1.5 --exec

This mode executes a workflow.

Normally, this switch is not called directly by the user. However, it is possible to execute exactly 1 workflow.

For this purpose, *kenaflow*.exe `--exec` is started in the workflow directory or the parameter `--workflow=<full-path>` is used with a complete path to a workflow configuration file, e.g.:

```
kenaflow.exe --exec --workflow=c:\workflows\wf1\_wfconfig.ps1
```

The option `--teststate=<state-code>[:<state-code>[:<state-code>...]]` can be used to allow only certain workflow steps for workflow execution. All steps not included here are ignored.

Execution delay means that for each workflow or additionally for each workflow step (for state machine workflows) you can specify how much time should elapse between two executions.

15.8.1.6 --teststate

Coming soon!

15.8.1.7 --notbe

The `--notbe` option can be used to disable all execution delays.

15.8.2 Workflows

15.8.2.1 Basics

15.8.2.1.1 --new [--wftype]

This allows you to create a new workflow.

The command must be executed in an **empty** directory.

You can specify `--wftype=<workflow-template>` with

- "statemachine"

- "list"
- "site" (default is "statemachine")

Example:

```
    --new --wftype=statemachine
```

15.8.2.2 HistoryList

These commands can be used to create, hide or display the history list according to the workflow configuration file in the current directory or the workflow configuration file specified with `--workflow=<full path>`.

- `--createhistorylist`
- `--hidehistorylist`
- `--showhistorylist`

15.8.2.3 DataList

These commands can be used to create, hide or display the "Workflow Data" list according to the workflow configuration file in the current directory or the workflow configuration file specified with `--workflow=<full-path>`.

- `--createdatalist`
- `--hidedatalist`
- `--showdatalist`

15.8.2.4 ConfigList

These commands can be used to create, hide or display the "Workflow Config" list according to the workflow configuration file in the current directory or the workflow configuration file specified with `--workflow=<full-path>`.

- `--createconfiglist`
- `--hideconfiglist`
- `--showconfiglist`

15.8.2.5 StateList

These commands can be used to create, hide or display the "Workflow Config" list according to the workflow configuration file in the current directory or the workflow configuration file specified with `--workflow=<full-path>`.

- `--createstatelist`
- `--hidestatelist`
- `--showstatelist`

15.8.2.6 Credentials User

These are used to store the credentials for SharePoint in a special credential file (`_wfconfig-cred.ps1`).

The workflow configuration file is stored in the same directory as `_wfconfig.ps1`. This is either in the current directory or is specified via `--workflow=<full-path>`.

- `--setusername`
- `--setpassword`

15.8.2.7 Credentials App

These are used to store the credentials for SharePoint Online in a special credential file (`_wfconfig-cred.ps1`).

The workflow configuration file is stored in the same directory as `_wfconfig.ps1`. This is either in the current directory or is specified via `--workflow=<full-path>`.

- `--appid`
- `--appsecret`

15.8.2.8 SharePoint Permissionsets

In the workflow configuration file, you can specify four permission sets that can be quickly and easily assigned to a user using the `Add-KFPermission`, `Remove-KFPermission`, and `Update-KFPermission` cmdlets.

We have compiled 4 meaningful SharePoint authorization levels that are often used. These follow the well-known CRUD model (Create, Read, Update, Delete). We have added "M" for "Manage" and "A" for "Approve".

Users or groups that have the corresponding permission set can perform very special tasks.

For example, anyone assigned "Create" may create elements, but not automatically read them. Accordingly, a user with "Read" authorization level may not automatically create elements.

You can specify names for the 4 authorization levels in the workflow configuration file.

With the program command `--createpermissionsets` these authorization levels are created in SharePoint and equipped with rights ("Base Permissions") according to our specifications.

Here is an overview of how the authorization levels are defined:

Create:

- Add list Items (AddListItems)
- Show List Items (ViewListItems)
- Create Alerts (CreateAlerts)
- View Pages (ViewPages)
- Open (Open)

Read:

- View list Items (ViewListItems)
- Open List Items (OpenItems)
- Show Version (ViewVersions)
- Show Pages (ViewPages)
- Open (Open)

Update:

- Edit List Items (EditListItems)

- Show List Items (ViewItems)
- View Pages (ViewPages)
- Open (Open)

Delete:

- Delete List Items (DeketeListItems)
- View List Items (ViewItems)
- Show Versions (ViewVersions)
- Delete Versions (DeleteVersions)
- Use APIs (UseRemoteAPIs)
- View Pages (ViewPages)
- Open (Open)

Manage:

- Edit list items (EditListItems)
- View list items (ViewItems)
- Approve list items (ApproveItems)
- Open list Items (OpenItems)
- Browse directories (BrowseDirectories)
- Show Versions (ViewVersions)
- Enumerate Permissions (EnumeratePermissions)
- Manage Permissions (ManagePermissions)
- Browse User Infos (BrowseUserInfo)
- Open (Open)

Approve:

- View list items (ViewItems)
- Approve list items (ApproveItems)
- Open list Items (OpenItems)
- Show Versions (ViewVersions)
- Browse directories (BrowseDirectories)
- Enumerate Permissions (EnumeratePermissions)
- Browse User Infos (BrowseUserInfo)
- Open (Open)

15.8.2.8.1 --createpermissionsets

This command creates or updates the defined permissions in SharePoint according to the workflow directory.

Please take care, that you have the required rights for this or execute this command under a user who has the required rights.

15.8.3 Tests

15.8.3.1 --sendmail="<mail-address>"

Can be used to test the sending of e-mails.

The <mail-address> is a valid e-mail address that can be reached via the SMTP server.

15.8.3.2 --testmain

This allows "multi-processing" to be tested. The "Main" process starts a "Client" process with which it communicates. The processes run independently, but exchange messages.

15.8.3.3 --rcfg

This program mode reads and loads all workflow configuration files in the workflow root directory defined in the global configuration or specified by the --workflowroot=<full-path> option.

The workflow configuration files are checked.

Changes to the workflow configuration files are monitored.

Thus central functions of the *kenaflow* runtime environment can be tested.

CHAPTER

SIXTEEN

GLOBAL CONFIGRUATION

The global configuration is stored in the *kenaflow* program directory (e.g. `C:\Program Files\kenaflow`).

filename: kenaflow.conf

In the standard configuration, this file is encrypted on the hard disk.

A mechanism provided by the Windows operating system is used for the encryption, which encrypts data using an automatically generated key ("Machine Key"). This key is valid for this Windows installation, but it is different on other Windows installations. This prevents `kenaflow.conf` from being copied from one computer to another.

The config file is saved as in binary serialized form because this is far more efficient than parsing XML on every start of a *kenaflow* process. The new serialized file `kenaflow.ser.conf` is created by the engine if:

- the engine was updated
- the configuration was modified (`--editconfig`)
- the "last write time" of the file `kenaflow.conf` has changed
- the "last write time" of the file `kenaflow.lib.dll` has changed.
- the path of the file has changed

If the serialized file is not valid it gets deleted if the account has permissions to do that.

The serialized file is encrypted with the data protection mechanism of Windows!

!!!! **Notice**: There is a property in the `kenaflow.conf` that can be used to disable encryption so that the global configuration file is stored unencrypted on the hard disk.

16.1 Structure

The configuration file is saved in the XML format.

The basic structure is as follows:

```
    <?xml version="1.0" encoding="utf-8"?>
    <config>
        <section name="<section-name>">
            <entry key="[key]" type="[typ]" comment="[kommentar]" default="[wert]">
↪[daten]</entry>
            <entry key="[key]" type="[typ]" comment="[kommentar]" default="[wert]">
↪[daten]</entry>
            <entry key="[key]" type="[typ]" comment="[kommentar]" default="[wert]">
↪[daten]</entry>
        </section>
    </config>
```

The division into sections (`<section/>`) is only included for reasons of clarity. Otherwise, it has no meaning and can be changed at will.

!!!! **Notice**: Sections cannot be nested!

! **Important**: XML comments that you store in the XML file are lost when you save them. Use the `comment` attribute.

Each configuration entry must have a unique key. This must not be used a second time in the same configuration file.

The type of configuration entry specifies the format in which the data of the configuration entry is expected.

- System.String
- System.Int32
- System.Int64
- System.Boolean
- System.Guid

The default value specifies the default setting specified by *kenaflow*. If no value is specified for the configuration parameter ("[data]"), then the default value is used.

The `comment` can be set arbitrarily. It is intended to document the settings.

The following sections and properties are defined:

- *Section "AfterSetup"*
- *Section "Licensing"*
- *Section "Basic"*
- *Section "Storage"*
- *Section "Monitoring"*
- *Section "Runtime"*
- *Section "Logging"*
- *Section "Main Mode"*
- *Section "SharePointConnection"*
- *Section "MultiProcessing"*
- *Section "Lock"*
- *Section "Mail"*
- *Section "MailDev"*
- *Section "RER"*
- *Section "cmdletConfic"*
- *Section "Links"*
- *Section "DoNotTouch"*

16.2 Complete file `kenaflow.conf`

Here you can find a copy of the complete file: *`kenaflow.conf`*

16.3 Section "AfterSetup

The section `AfterSetup` contains the properties that **must** be set by the administrator after installation.

16.3.1 adminMails

```
        <entry key="adminMails" type="System.String" comment="split by ;"></entry>
```

Zero, one or more semicolon separated administrator e-mail addresses.

This will be used to inform administrators about errors in the workflow engine.

16.3.2 WorkflowRoot

```
        <entry key="WorkflowRoot" type="System.String" default=""></entry>
```

Workflow root directory.

Full path to an existing directory. The actual workflows are located in its subdirectories.

16.3.3 encryptionPassphrase

```
        <entry key="encryptionPassphrase" type="System.String" comment="do not␣
↪change after first usage." default="kenafl0W!"></entry>
```

Encryption Passphrase. It will be used to encrypt:

- Remote Events
- Data protected with `[Protect-KFData]` and `[Unprotect-KFData]`
- Serialized Workflow Configuration files.

! This is not allowed to be changed after the first run! You need to configure it with a strong password after the setup!

16.3.4 activeDirectoryDomain

```
        <entry key="activeDirectoryDomain" type="System.String" comment="please␣
↪specify your domain as FQDN" default="."></entry>
```

The Windows Active Directory domain the engine is running in.

16.3.5 instanceName

= *kenaflow* 4.0.22

```
        <entry key="instanceName" type="System.String" comment="if not specified␣
↪its the instance id" default="">kenaflow-debug-instance</entry>
```

With this option it is possible to specify a name for the instance of *kenaflow*.

This can be used in scripts with predefined variable `$KFInstanceName`.

16.3.6 Full Section Default

```
    <section name="aftersetup">
        <entry key="adminMails" type="System.String" comment="split by ;"></entry>
        <entry key="WorkflowRoot" type="System.String" default=""></entry>
        <entry key="encryptionPassphrase" type="System.String" comment="do not␣
↪change after first usage." default="kenafl0W!"></entry>
        <entry key="activeDirectoryDomain" type="System.String" comment="please␣
↪specify your domain as FQDN" default="."></entry>
        <entry key="instanceName" type="System.String" comment="if not specified␣
↪its the instance id" default="">kenaflow-debug-instance</entry>
    </section>
```

16.3.7 Related Cmdlets

16.4 Section “Licensing

In this section you will find information about licensing, activation and activation of *kenaflow*.

16.4.1 licensingMailAddress

```
        <entry key="licensingMailAddress" type="System.String" default=""></entry>
```

Each *kenaflow* license and its serial is bound to an email address. This is stored in this setting during activation.

16.4.2 Serial

```
        <entry key="Serial" type="System.String"></entry>
```

You will receive the serial number when you purchase the software.

Each serial number is unique.

The serial number does not indicate whether it is a full version or a free version.

16.4.3 ActivationCode

```
        <entry key="ActivationCode" type="System.String"></entry>
```

This setting contains the activation code after it was first retrieved with `-getactivationcode`.

16.4.4 lastActivationCode

```
        <entry key="lastActivationCode" type="System.String" default=""></entry>
```

After a successful activation of *kenaflow* the used activation code is stored here.

The activation code could change when changing virtual or physical hardware. So it’s maybe necessary to access the last used activation code.

16.4.5 lastOffineDeactivationCode

```
        <entry key="lastOffineDeactivationCode" type="System.String" default=""></
↪entry>
```

When deactivated, a code is generated. This code is stored here. The next time it is activated, the setting is deleted.

16.4.6 Full Section Default

```
    <section name="licensing">
        <entry key="licensingMailAddress" type="System.String" default=""></entry>
        <entry key="Serial" type="System.String"></entry>
        <entry key="ActivationCode" type="System.String"></entry>
        <entry key="lastActivationCode" type="System.String" default=""></entry>
        <entry key="lastOffineDeactivationCode" type="System.String" default=""></
↪entry>
    </section>
```

16.5 Section “Basic

In this section you will find basic settings of *kenaflow*. Some of them may not be changed. These are marked in red.

16.5.1 database

```
        <entry key="database" type="System.String" comment="SQLite database␣
↪location" default=".\kenaflow.sqlite"></entry>
```

Optionally *kenaflow* can store workflow generated data in an SQLite database.

Here you can specify the path. If its a relative path the database is stored below the _kenaflow ‘ProgramData’ directory.

16.5.2 globalFloodProtectionFile

```
        <entry key="globalFloodProtectionFile" type="System.String" default=".\
↪adminAlertFloodProtection.txt"></entry>
```

When *kenaflow* sends error mails to administrators it’s possible the a huge amout of mails would be send without flood protection.

In this setting the file is specified where *kenaflow* stores the date and time of the last send error mail for administrators.

16.5.3 globalFloodProtectionTBE

```
        <entry key="globalFloodProtectionTBE" type="System.Int32" default="300"␣
↪comment="in seconds"></entry>
```

Together with setting `globalFloodProtectionFile` above this is used to prevent huge amount of error mails for administrators.

16.5.4 globalWorkflowExecutionTBE

```
        <entry key="globalWorkflowExecutionTBE" type="System.Int32" default="300"␣
↪comment="in seconds"></entry>
```

Workflows are normally executed on a regular basis. A certain amount of time must elapse between the start of *one* execution and the start of *next* execution. This is specified here as the default for all workflows that do not have their own settings.

16.5.5 globalWorkflowStateTBE

```
        <entry key="globalWorkflowStateTBE" type="System.Int32" default="60"␣
↪comment="in seconds"></entry>
```

A separate time can be defined for each workflow status, which must lie between two executions of the status. The time in seconds between the start of *one* execution and the start of *next* execution is specified.

16.5.6 ConfigEditor

```
        <entry key="ConfigEditor" type="System.String" default="notepad.exe"></
↪entry>
```

If this setting is empty, Windows Notepad is used when editing the configuration file. Here you can specify a path to an alternative editor. The path must be absolute. The name of the configuration file is passed as the only parameter at startup.

We have seen that there are problems with some editors if they are already open when the `kenaflow.exe --editconfig` command is executed. For example, NotePad++ does not open a new process but passes the file to be opened to the already open process. This is not recognized by *kenaflow*.

When editing the *encrypted configuration file* `kenaflow.conf` a temporary file is created with the decrypted data from the configuration file. This temporary file is opened in the editor. This file is read and deleted after closing the editor (termination of the process). The new data is stored encrypted again.

16.5.7 EncryptConfig

```
        <entry key="EncryptConfig" type="System.Boolean" default="true"></entry>
```

Specifies whether the configuration file `kenaflow.conf` should be encrypted. If "TRUE", then it is encrypted. If "FALSE", it's not. This can be changed as required and takes effect immediately when the editor is closed.

!!! This setting is not available in FREE editon and DEVELOPER edition of *kenaflow*!

16.5.8 Branch

```
        <entry key="Branch" type="System.String" default="release-4-0" comment=
↪"for updates..."></entry>
```

The branch specifies from which source *kenaflow* is obtained and updated. The branch may only be changed if you have been asked to do so.

Normally this is changed before doing an version upgrade, e.g. from "release-4-0" to "release-5-0".

!! The default setting depends on the kenflow edition you have installed:

- release-4-0
- free-4-0
- developer-4-0

"release-4-0" is used for *kenaflow Standard Edition*!

Changing the value from "free-4-0" to "release-4-0" will cause *kenaflow* to fail **if** you do not upgrade the license from FREE to STANDARD!

16.5.9 InfoServerPort

```
        <entry key="InfoServerPort" type="System.Int32" default="-1"></entry>
```

!! This is part of an experimental funtion. Please do not change it.

16.5.10 ProgramDataFolder

```
        <entry key="ProgramDataFolder" type="System.String" default=""></entry>
```

Here you can specify an absolute path under which *kenaflow* stores data.

The service account executing *kenaflow* must have read, write, change and delete permissions under that path.

Please see *Advanced Configuration*

!!! This setting is not available in FREE editon and DEVELOPER edition of *kenaflow*!

16.5.11 CustomLibrary

```
        <entry key="CustomLibrary" type="System.String" default="" comment=
↪"relative to program folder"></entry>
```

Here you can specify a path to a global PowerShell library that is loaded and executed before each item, mail or workflow processing.

Here you can for example store global functions.

16.5.12 VSCodeDir

```
        <entry key="VSCodeDir" type="System.String" default=".\vscode" comment=
↪"please use the version of Visual Studio Code that we supply."></entry>
```

kenaflow contains a local copy of Visual Studio Code. This can be used to develop workflows with PowerShell and works well for *kenaflow* Debugging.

Please read *Debugging*

Visual Studio Code is started using `kenaflow.exe --vscode`

If you want to specify you own installation of Visual Studio code than you need to specify the *path* of VSCode here.

16.5.13 encryptSerializedWFCONFIG

```
        <entry key="encryptSerializedWFCONFIG" type="System.Boolean" default="true
↪"></entry>
```

For performance reasons the configuration file `_wfconfig.ps1` of each workflow is serialized as binary data stream.

For security reasons, this serialized file is encrypted.

Here this encryption can be switched off.

16.5.14 allowPowerShellPlaceholders

```
        <entry key="allowPowerShellPlaceholders" type="System.Boolean" default=
↪"false"></entry>
```

kenaflow templates () maybe contain PowerShell enabled placeholders marked with << and >>. To enable this the administrator must set this setting to `true`.

By default it is disabled!

16.5.15 PluginsEnabled

```
        <entry key="PluginsEnabled" type="System.Boolean" default="true"></entry>
```

This enables or disables the plugin subsystem of *kenaflow*.

16.5.16 PluginsFolder

```
        <entry key="PluginsFolder" type="System.String" default=".\Plugins"␣
↪comment="relative path to program folder or absolute path"></entry>
```

This tells *kenaflow* where to find plugin assemblies.

If this path is relative its suspected to resist below the program folder where `kenaflow.exe` is located.

16.5.17 httpSecurityProtocolType

```
      <entry key="httpSecurityProtocolType" type="System.String" default="Tls12"␣
↪comment="Values:Tls12,Tls,Tls11,Ssl3 (sperated by comma)"></entry>
```

New since 4.0.0.

This setting can be used to set the HTTP security protocol type(s) for *kenaflow*. It specifies what security protocol types can be used by *kenaflow*.

Under the hood it sets the static property `System.Net.ServicePointManager.SecurityProtocol` to the specified value(s).

Valid values are:

- Tls12
- Tls
- Tls11
- Ssl3

Default is `Tls12`. This is one and only security protocol type that should be used *today*.

16.5.18 Full Section Default

```
    <section name="basic">
      <entry key="database" type="System.String" comment="SQLite database␣
↪location" default=".\kenaflow.sqlite"></entry>
      <entry key="globalFloodProtectionFile" type="System.String" default=".\
↪adminAlertFloodProtection.txt"></entry>
      <entry key="globalFloodProtectionTBE" type="System.Int32" default="300"␣
↪comment="in seconds"></entry>
      <entry key="globalWorkflowExecutionTBE" type="System.Int32" default="300"␣
↪comment="in seconds"></entry>
      <entry key="globalWorkflowStateTBE" type="System.Int32" default="60"␣
↪comment="in seconds"></entry>
      <entry key="ConfigEditor" type="System.String" default="notepad.exe"></
↪entry>
      <entry key="EncryptConfig" type="System.Boolean" default="true"></entry>
      <entry key="Branch" type="System.String" default="release-4-0" comment=
↪"for updates..."></entry>
      <entry key="InfoServerPort" type="System.Int32" default="-1"></entry>
      <entry key="ProgramDataFolder" type="System.String" default=""></entry>
      <entry key="CustomLibrary" type="System.String" default="" comment=
↪"relative to program folder"></entry>
      <entry key="VSCodeDir" type="System.String" default=".\vscode" comment=
↪"please use the version of Visual Studio Code that we supply."></entry>
      <entry key="encryptSerializedWFCONFIG" type="System.Boolean" default="true
↪"></entry>
      <entry key="allowPowerShellPlaceholders" type="System.Boolean" default=
↪"false"></entry>
      <entry key="PluginsEnabled" type="System.Boolean" default="true"></entry>
      <entry key="PluginsFolder" type="System.String" default=".\Plugins"␣
↪comment="relative path to program folder or absolute path"></entry>
      <entry key="httpSecurityProtocolType" type="System.String" default="Tls12"␣
↪comment="Values:Tls12,Tls,Tls11,Ssl3 (sperated by comma)"></entry>
    </section>
```

16.6 Section “Storage

In this section you will find storage related settings of *kenaflow*.

“Storage” means that you can use the following cmdlets to store and retrieve binara data or serialized objects to/from disk storage. Please read article *Storage*.

16.6.1 storageEnabled

```
        <entry key="storageEnabled" type="System.Boolean" default="true"></entry>
```

With this setting the storage function can be disabled globally.

!!! If disabled but workflows try to use the storage cmdlets an exception is thrown.

16.6.2 storagePath

```
        <entry key="storagePath" type="System.String" default=".\storage"></entry>
```

This setting specifies the storage path. The *kenaflow* execution account and every workflow developer user account must have permissions to

- Create
- Read
- Modify
- Delete files and folders in the directory.

16.6.3 storageMaxSize

```
        <entry key="storageMaxSize" type="System.Int32" default="1073741824"␣
↪comment="1GB"></entry>
```

This setting can be used to control the maximum size of data that can be written to the storage.

16.6.4 Full Section Default

```
      <section name="storage">
        <entry key="storageEnabled" type="System.Boolean" default="true"></entry>
        <entry key="storagePath" type="System.String" default=".\storage"></entry>
        <entry key="storageMaxSize" type="System.Int32" default="1073741824"␣
↪comment="1GB"></entry>
      </section>
```

16.7 Section “Monitoring

The settings in this section control the monitoring capabilities of *kenaflow*.

For details please read article *Monitoring*

16.7.1 usePerformanceCounter

```
        <entry key="usePerformanceCounter" type="System.Boolean" default="false"></
↪entry>
```

kenaflow offers a basic set of performance counters that integrate into Windows OS and can be used locally on the *kenaflow* server using `perfmon` or with any third party monitoring that can leverage Windows performance counter.

With this setting the use of performance counters can be enabled or disabled (default).

16.7.2 WorkReportHookAddress

```
        <entry key="WorkReportHookAddress" type="System.String" default=""></entry>
```

kenaflow is able to report its work to a webhook for monitoring purpose.

With this setting you can specify a webhook that is able to receive *kenflow* work reports.

16.7.3 ClientStartReportHookAddress

```
        <entry key="ClientStartReportHookAddress" type="System.String" default="">
↪</entry>
```

kenaflow uses sub processes for workflow execution and several other tasks.

Sub processes are started and/or restarted regulary.

This web hook can be used for monitoring purpose to inform a monitoring system about the start of sub processes.

16.7.4 PingICMP

```
        <entry key="PingICMP" type="System.String" comment="[ip/host]|[time-
↪between]|[max-ok-latency-msec]|[stop-after-failed-count]|return-to-normal-after-
↪count]~..."></entry>
```

kenaflow is able to monitor other network related devices to check its connectivity.

Here you can specify zero, one or more devices that have to be available over ICMP ping to infornm kenaflow about network connectivity.

One entry consists of this settings, separated by | (pipe character):

- IP address or host name
- time between two ICMP pings (in milliseconds)
- maximum latency (in milliseconds)
- mark this entry (device) as “failed” (= not reachable / disconnected) after this number of failures
- mark this entry (device) as “normal” (= reachable / connected) after this number of successful ping AFTER it’s in “failed” state

Sample for two entries: 172.16.1.1|5000|1|5|5~domaincontroller2.company.local|5000|1|5|5

16.7.5 PingWebHook

```
        <entry key="PingWebHook" type="System.String" comment="[url]|[time-
↪between]|[expected-http-status-code]|[stop-after-failed-count]|return-to-normal-
↪after-count]~..."></entry>
```

kenaflow is able to monitor other network related devices to check its connectivity.

Here you can specify zero, one or more devices that have to be available over HTTP web request to infornm kenaflow about network connectivity.

One entry consists of this settings, separated by | (pipe character):

- URL to web resource
- time between two HTTP request (in milliseconds)
- desired HTTP response status code
- mark this entry (device) as "failed" (= not reachable / disconnected) after this number of failures
- mark this entry (device) as "normal" (= reachable / connected) after this number of successful ping AFTER it's in "failed" state

Sample for two entries: http://ping1.company.local|5000|200|5|5~https://172.16.1.1|5000|200|5|5~

16.7.6 StopCountFailedPings

```
        <entry key="StopCountFailedPings" type="System.Int32" comment="how much␣
↪ping conditions must fail from setting above before pausing the engine?" default=
↪"1"></entry>
```

The settings determine the number of ping tests that must fail before *kenaflow* stops work.

It must be >= 1.

It is not used if there is no setting for `PingICMP` and/or `PingWebHook`.

16.7.7 ErrorWebHook

```
        <entry key="ErrorWebHook" type="System.String" comment="[url]|[level:␣
↪(+)0=waring, 1=error, 2 = critical]|['get' or 'post']~...">http://127.0.0.1:3000/
↪|+1|post</entry>
```

In case of (critical) errors or warnings this web hook setting can be used to contact a IT monitoring system.

Such a web hook is defined by an URL, a log level and a HTTP method.

The URL ist the address of your monitoring system.

The log level defines the level of messages that have to be reported:

- `0` = Warning
- `1` = Error
- `2` = Critical Error

Using + before the level the level ist interpreted as "at least" / >=.

Example:

- `+1` means `Error or Critical Error`.

- `0` (without +) means: `Warning`

The method can be used to send error data or just a ping with no data.

Method:

- `GET`: just ping the web hook
- `POST`: send error data to the web hook

The sended data contains information about the error. The following information is separated by `CRLF` characters

- Timestamp in format `yyyy-MM-dd HH:mm:ss`
- LogLevel as string
- Process Id (Windows process of `kenaflow.exe`)
- Thread Id
- Message

The message can have multiple lines that are separated by `CRLF`.

Sample web hook:

```
    http://127.0.0.1:3000/|+1|post
```

Multiple hooks can be defined by separating them using `~`.

Example: You could have separate hooks for 1) warnings and 2) (critical) errors.

```
    http://127.0.0.1:3000/|0|post~http://127.0.0.1:4000/|+1|post
```

16.7.8 FreeDiskSpaceDrives

```
        <entry key="FreeDiskSpaceDrives" type="System.String" comment="example:
↪'C:;D:'. separated by ;" default="">C:;R:</entry>
```

This setting can be used to enable free disk space monitoring.

One or more drives to monitor can be specified here.

if the available memory of the specified drives is below the amount of memory as specified in `FreeDiskSpaceThreshold` *kenaflow* will not start new sub processes.

Already running child processes will terminate themselves if the free disk space is below `FreeDiskSpaceThreshold` for more than 5 minutes.

The main process of *kenaflow* (started with `--run`) will not terminate but does not start any sub processes.

16.7.9 FreeDiskSpaceThreshold

```
        <entry key="FreeDiskSpaceThreshold" type="System.Int32" comment="in␣
↪megabyte!" default="500"></entry>
```

If drives are specified in `FreeDiskSpaceDrives` and this feature is activated the free space threshold is specified in this setting.

Amount of free disk space in MEGABYTES!

16.7.10 FreeDiskSpaceWebHook

```
        <entry key="FreeDiskSpaceWebHook" type="System.String" comment="[url]|~...
↪">http://127.0.0.1:3000/</entry> <!--http://127.0.0.1:3000/-->
```

This setting can be used to ping one or more web hook(s) in case of free space is running out. The ping is send every 30 seconds while free disk space is below `FreeDiskSpaceThreshold`.

Currently there is no info send about the drive that is running out of free disk space.

Example for hook

```
    http://127.0.0.1:3000/
```

Multiple web hooks can be separated by `;`

Example for two wb wooks:

```
    http://127.0.0.1:3000/~http://127.0.0.1:4000/
```

16.7.11 Full Section Default

```
      <section name="monitoring">
        <entry key="usePerformanceCounter" type="System.Boolean" default="false"></
↪entry>
        <entry key="WorkReportHookAddress" type="System.String" default=""></entry>
        <entry key="ClientStartReportHookAddress" type="System.String" default="">
↪</entry>
        <entry key="PingICMP" type="System.String" comment="[ip/host]|[time-
↪between]|[max-ok-latency-msec]|[stop-after-failed-count]|return-to-normal-after-
↪count]~..."></entry>
        <entry key="PingWebHook" type="System.String" comment="[url]|[time-
↪between]|[expected-http-status-code]|[stop-after-failed-count]|return-to-normal-
↪after-count]~..."></entry>
        <entry key="StopCountFailedPings" type="System.Int32" comment="how much␣
↪ping conditions must fail from setting above before pausing the engine?" default=
↪"1"></entry>
        <entry key="ErrorWebHook" type="System.String" comment="[url]|[level:␣
↪(+)0=waring, 1=error, 2 = critical]|['get' or 'post']~...">http://127.0.0.1:3000/
↪|+1|post</entry>
        <entry key="FreeDiskSpaceDrives" type="System.String" comment="example:
↪'C:;D:'. separated by ;" default="">C:;R:</entry>
        <entry key="FreeDiskSpaceThreshold" type="System.Int32" comment="in␣
↪megabyte!" default="500"></entry>
        <entry key="FreeDiskSpaceWebHook" type="System.String" comment="[url]|~...
↪">http://127.0.0.1:3000/</entry> <!--http://127.0.0.1:3000/-->
      </section>
```

16.8 Section “Runtime

16.8.1 Settings for all workflow types

16.8.1.1 MaxExecutionTime

```
        <entry key="MaxExecutionTime" type="System.Int32" default="600"></entry>
```

Specifies how long a workflow should be executed. When the time has expired, the workflow asked to stop.

This setting can be overwritten for each workflow or workflow step.

If a workflow step has a larger setting than the workflow itself, the workflow is not terminated during the workflow step execution.

16.8.1.2 defaultScriptRunTimeout

```
        <entry key="defaultScriptRunTimeout" type="System.Int32" default="600"␣
↪comment="in seconds"></entry>
```

Workflow scripts are executed inside PowerShell. As in every programming language it is possible in PowerShell to create loops. It is not possible for *kenaflow* to check the "code quality" of a workflow script. If you (accidentally) create an endless loop *kenaflow* would not be able to stop the script. Therefore it has an security feature that forces scripts to stop if they run too long.

This setting can be overwritten by each workflow.

In the script it is possible to manipulate the runtime:

- *Set-KFScriptRunTimeout*
- *Get-KFScriptRunTimeout*

16.8.1.3 MaxRuntimeOfChildProcess

```
        <entry key="MaxRuntimeOfChildProcess" type="System.Int32" default="86400"␣
↪comment="Time in Sec. before a child process is getting restarted - for security␣
↪reasons."></entry>
```

Processes such as "remote event runner" or "global remote event receiver" are stated by the main process of *kenaflow* that controls the multi process execution of the *kenaflow* instance.

For stability reasons it's recommanded to restart such "long running" child processes after a certain amount of time.

With this setting you can control the maximum runtime, before the child process shuts *itself* down. With the setting `VariationOfMaxRuntimeOfChildProcess` you can vary this maximum execution time so that not every child process shuts itself down at the same moment = amout of time.

16.8.1.4 VariationOfMaxRuntimeOfChildProcess

```
        <entry key="VariationOfMaxRuntimeOfChildProcess" type="System.Int32"␣
↪default="14400" comment="Time in Sec. before a child process is getting␣
↪restarted - for security reasons."></entry>
```

Please read the description of setting `MaxRuntimeOfChildProcess`.

16.8.1.5 poshQueueSize

```
<entry key="poshQueueSize" type="System.Int32" default="5"></entry>
```

Available since *kenaflow* 4.0.12.

Remote events are executed in special *kenaflow* processes that are always running in background, waiting for incoming remote events.

These "remote runner" processes hold prepared PowerShell runtimes for remote event execution.

With this setting you can specify the amount of prepared runtimes.

When starting the remote event execution such a prepared runtime gets initialized with the workflow the remote event belongs to.

If a prepared runtime is used a new one is prepared and added to the queue of prepared runtimes.

After the execution of a remote event the PowerShell runtime session is destroyed.

16.8.1.6 culture

```
<entry key="culture" type="System.Int32" default="1033" comment="This␣
↪culture overwrites the users culture. Set to -1 to disable."></entry>
```

This can be used to set the culture of *kenaflow* scripts. It is used in automatic mode and in debug mode.

The list of LCIDs can be found here:

If set to `-1` this feature is disabled. The users culture is used.

This solves a problem that this command could fail, e.g. if debug user has "German" locale and the execution account has "English": `[DateTime]::Parse('12/31/2019')`

16.8.2 Settings related to SharePoint workflows (site workflow, list workflow, state machine worklfow)

16.8.2.1 ItemReCheckTime

```
<entry key="ItemReCheckTime" type="System.Int32" default="60"></entry>
```

SharePoint is constantly in use through users and processes such as workflows. List items change. *kenaflow* uses queries to determine the list items to be processed.

If the query has taken place, it is not executed again until all determined list items have been processed. If the elements change in between, they still appear in the query result that was fetched before.

With the setting "ItemReCheckTime" you can set the point in time, starting at the time of first querying items, from which each list item is to be checked again to determine whether processing is still necessary.

This setting can be overwritten for each workflow or workflow step.

16.8.2.2 ItemBatchCount

```
        <entry key="ItemBatchCount" type="System.Int32" default="1000"></entry>
```

List items for processing are fetched from SharePoint using queries. They are gathered in batches of the size, specified in this setting.

!! CAUTION: This setting must not collide with the ListitemThreshold set in SharePoint. This number must be smaller. The default setting of SharePoint is 5000, which should not be exceeded. 1000 is optimal.

The setting "ItemReCheckTime" should also be observed. This setting can be overwritten for each workflow or workflow step.

16.8.2.3 UserProfileCacheLifetime

```
        <entry key="UserProfileCacheLifetime" type="System.Int32" default="60"␣
↪comment="in seconds"></entry>
```

To get informations about users from the User Profile Service Application of SharePoint, you can use the cmdlet *Get-KFUserInfo*.

This information is cached in the currently running *kenaflow* process that executes the current workflow.

The informations in the User Profile change from time to time but nomally not in seconds after you query them. Therefore *kenaflow* uses a caching to reduce the communication overhead to SharePoint.

You can reduce the default cache lifetime for user informations here.

This can be overwritten in each workflow.

16.8.2.4 Full Section Default

```
      <section name="runtime">
        <entry key="ItemReCheckTime" type="System.Int32" default="60"></entry>
        <entry key="ItemBatchCount" type="System.Int32" default="1000"></entry>
        <entry key="MaxRuntimeOfChildProcess" type="System.Int32" default="86400"␣
↪comment="Time in Sec. before a child process is getting restarted - for security␣
↪reasons."></entry>
        <entry key="VariationOfMaxRuntimeOfChildProcess" type="System.Int32"␣
↪default="14400" comment="Time in Sec. before a child process is getting␣
↪restarted - for security reasons."></entry>
        <entry key="MaxExecutionTime" type="System.Int32" default="600"></entry>
        <entry key="UserProfileCacheLifetime" type="System.Int32" default="60"␣
↪comment="in seconds"></entry>
        <entry key="defaultScriptRunTimeout" type="System.Int32" default="600"␣
↪comment="in seconds"></entry>
        <entry key="culture" type="System.Int32" default="1033" comment="This␣
↪culture overwrites the users culture. Set to -1 to disable."></entry>
            <entry key="poshQueueSize" type="System.Int32" default="5"></entry>
      </section>
```

16.9 Section “Logging

16.9.1 useSerilog

= *kenaflow* 4.0.23

```
        <entry key="useSerilog" type="System.Boolean" default="false"></entry>
```

Can be used to enable the Serilog based logging.

If Serilog is enabled the “old” logging mechanism / format is disabled. The Serilog log files cannot be read with Ulsviewer!

16.9.2 serilogBaseTemplateConsole

= *kenaflow* 4.0.23

```
        <entry key="serilogBaseTemplateConsole" type="System.String" default="[
↪{Level}] {Message}{NewLine}{Exception}"></entry>
```

“Output template” for the Serilog Console Sink.

16.9.3 serilogBaseTemplateFile

= *kenaflow* 4.0.23

```
        <entry key="serilogBaseTemplateFile" type="System.String" default="[{Level}
↪] {Message}{NewLine}{Exception}"></entry>
```

“Output template” for the Serilog File Sink.

16.9.4 serilogTemplate

= *kenaflow* 4.0.23

```
        <entry key="serilogTemplate" type="System.String" default="{TS:HH:mm:ss}
↪{Process} {Event} {ThreadId} {Correlation} {Text:l}"></entry>
```

16.9.5 minConsoleLogLevel

```
        <entry key="minConsoleLogLevel" type="System.Int32" default="2" comment=
↪"0=debug, 1=output, 2=verbose, 3=warning, 4=normal, 5=error"></entry>
```

“0=debug, 1=output, 2=verbose, 3=warning, 4=normal, 5=error”

Specifies the severity of log messages that are shown in the console window - if it is visible.

Possible Values:

| Value | Severity | | - | - | | 0 | Debug | | 1 | Output | | 2 | Verbose | | 3 | Warning | | 4 | Normal | | 5 | Error |

16.9.6 minFileLogLevel

```
        <entry key="minFileLogLevel" type="System.Int32" default="0" comment=
↪"0=debug, 1=output, 2=verbose, 3=warning, 4=normal, 5=error"></entry>
```

Specifies the severity of log messages that are written to the log file.

| Value | Severity | | - | - | | 0 | Debug | | 1 | Output | | 2 | Verbose | | 3 | Warning | | 4 | Normal | | 5 | Error |

16.9.7 dontLogIfMainWindowIsHidden

```
        <entry key="dontLogIfMainWindowIsHidden" type="System.Boolean" default=
↪"true" comment="if the kenaflow process window is hidden no log info is written␣
↪to the console"></entry>
```

This disables the log output to the console window if it is not visible.

This happens if the Windows session of the user is non-interactive - e.g. for a service account - or if the console window was minimized.

If this setting is “TRUE” and the console window is hidden log messages will not be written to the console window for performance reasons.

16.9.8 logDir

```
        <entry key="logDir" type="System.String" default=".\log"></entry>
```

Specifies the directory where the logs are stored. For relative path specifications, the “ProgramDataFolder” is used as root.

16.9.9 logEnabled

```
        <entry key="logEnabled" type="System.Boolean" default="true"></entry>
```

Indicates whether the log is activated. If “FALSE”, no log files are written to the hard drive or to the console window.

16.9.10 logOnlyMainProcess

```
        <entry key="logOnlyMainProcess" type="System.Boolean" default="true"></
↪entry>
```

Specifies whether only the main process writes log files to the hard disk or also subprocesses that are started by the main process.

16.9.11 logMemMaxAge

```
        <entry key="logMemMaxAge" type="System.Int32" default="3600" comment=
↪"seconds"></entry>
```

Log messages that are displayed in the console remain in the memory of the *kenaflow* processes. Here you can specify after how many seconds these data can be discarded. Only the oldest data is deleted.

16.9.12 logFileMaxAge

```
        <entry key="logFileMaxAge" type="System.Int32" default="14" comment="days">
↪</entry>
```

Specifies the number of days that the log files are to be kept.

16.9.13 logCleanupRunner

```
        <entry key="logCleanupRunner" type="System.Int32" default="120" comment=
↪"in seconds. Amount of time before cleanup runs."></entry>
```

Log messages are queued in memory before they are displayed in the console window (if the process is visible) and before they are written to disk (if enabled).

There is a sub process that cleans up the used memory. This setting here specifies how often the cleanup should run.

16.9.14 logCleanupRunner

```
        <entry key="logfileCleanupRunner" type="System.Int32" default="3600"␣
↪comment="in seconds. Amount of time before log file cleanup runs."></entry>
```

Log files on disk will grow and can be large! 200MB per hour are more than possible! Here you specify how often *kenaflow* checks old log files on disk. It will remove old log files that are older than `logFileMaxAge` days.

16.9.15 logFCE

```
        <entry key="logFCE" type="System.Boolean" default="false"></entry>
```

This can be used to determine whether "First Chance Exceptions" should be logged. These are exceptions in the program code that occur but do not necessarily represent errors. An exception handled in the program code is no cause for concern. Therefore, the storage of this information is normally superfluous.

16.9.16 logOutput

```
        <entry key="logOutput" type="System.Boolean" default="false"></entry>
```

Objects in the PowerShell pipeline that reach the top execution level of *kenaflow* (outside workflow scripts) can be serialized to the log.

16.9.17 sendOutputToMainProcess

```
        <entry key="sendOutputToMainProcess" type="System.Boolean" default="false">
↪</entry>
```

PowerShell cmdlets often return values. Normally these return values are stored in variables. But if there is no "receiver" for the data it is written to the PowerShell pipeline. *kenaflow* will get this data on the last level of PowerShell execution.

This setting here specifies whether this return values (objects) are shown in the log or not.

16.9.18 logfileLockTimeout

```
        <entry key="logfileLockTimeout" type="System.Int32" default="500" comment=
↪"in milliseconds"></entry>
```

Each process tries to write log messages to the unified log file. This is handled by a subsystem of each kenaflow process and should not take long. During write the log file is locked.

The setting here specifies how long the log-writing sub system should wait for the lock before doing other things. If is really fails the process will keep the log in memory and will retry to write it later.

16.9.19 logCleanupThreadLogBlocks

```
        <entry key="logCleanupThreadLogBlocks" type="System.Int32" default="60"␣
↪comment="in seconds. Amount of time before cleanup runs."></entry>
```

Functional parts of the software are grouped into logical blocks. *kenaflow* tries to group log messages in the same way, e.g. for detailed log analysis.

Each logical block gets a number, assigned at runtime. A sub system cleanup this up. The setting here specifies the amount of time (in seconds) between two cleanup attempts.

16.9.20 heartbeatMessageInLog

```
        <entry key="heartbeatMessageInLog" type="System.Boolean" default="false"></
↪entry>
```

If enabled on each heartbeat communication a message is written to the log with level 'Debug'.

16.9.21 garbageCollectionMessageInLog

```
        <entry key="garbageCollectionMessageInLog" type="System.Boolean" default=
↪"false"></entry>
```

If enabled on each .NET garbage collection cleanup request a message is written to the log with level 'Debug'.

16.9.22 disableStacktrace

```
        <entry key="disableStacktrace" type="System.Boolean" default="false"></
↪entry>
```

If enabled the .net Stack Trace is included in exception log messages.

16.9.23 useMailHashing

= *kenaflow* 4.0.23

```
        <entry key="useMailHashing" type="System.Boolean" default="true"></entry>
```

This enables / disables the hash based flood protection for error mails.

16.9.24 mailhashDir

= *kenaflow* 4.0.23

```
        <entry key="mailhashDir" type="System.String" default=".\mailHash"></entry>
```

Folder for mail hash files. Full path or relative path to `ProgramDataFolder` setting in global configuration.

16.9.25 mailhashIntervals

= *kenaflow* 4.0.23

```
        <entry key="mailhashIntervals" type="System.String" default="5,10,15,20,30,
↪35,60" comment="comma separated list of minutes between identical mails"></entry>
```

A comma separated list of minute intervals.

Each value must be greater to the previous one!

16.9.26 Full Section Default

```
      <section name="logging">
        <entry key="useSerilog" type="System.Boolean" default="false"></entry>
        <entry key="serilogBaseTemplateConsole" type="System.String" default="[
↪{Level}] {Message}{NewLine}{Exception}"></entry>
        <entry key="serilogBaseTemplateFile" type="System.String" default="[{Level}
↪] {Message}{NewLine}{Exception}"></entry>
        <entry key="serilogTemplate" type="System.String" default="{TS:HH:mm:ss}
↪{Process} {Event} {ThreadId} {Correlation} {Text:l}"></entry>
        <entry key="minConsoleLogLevel" type="System.Int32" default="2" comment=
↪"0=debug, 1=output, 2=verbose, 3=warning, 4=normal, 5=error"></entry>
        <entry key="minFileLogLevel" type="System.Int32" default="0" comment=
↪"0=debug, 1=output, 2=verbose, 3=warning, 4=normal, 5=error"></entry>
        <entry key="dontLogIfMainWindowIsHidden" type="System.Boolean" default=
↪"true" comment="if the kenaflow process window is hidden no log info is written␣
↪to the console"></entry>
        <entry key="logDir" type="System.String" default=".\log"></entry>
        <entry key="logEnabled" type="System.Boolean" default="true"></entry>
        <entry key="logOnlyMainProcess" type="System.Boolean" default="true">></
↪entry>
        <entry key="logMemMaxAge" type="System.Int32" default="3600" comment=
↪"seconds"></entry>
```

(continues on next page)

(continued from previous page)

```
        <entry key="logFileMaxAge" type="System.Int32" default="14" comment="days">
↪</entry>
        <entry key="logCleanupRunner" type="System.Int32" default="120" comment=
↪"in seconds. Amount of time before cleanup runs.">60</entry>
        <entry key="logfileCleanupRunner" type="System.Int32" default="3600"␣
↪comment="in seconds. Amount of time before log file cleanup runs.">60</entry>
        <entry key="logFCE" type="System.Boolean" default="false"></entry>
        <entry key="logOutput" type="System.Boolean" default="false"></entry>
        <entry key="sendOutputToMainProcess" type="System.Boolean" default="false">
↪</entry>
        <entry key="logfileLockTimeout" type="System.Int32" default="500" comment=
↪"in milliseconds"></entry>
        <entry key="logCleanupThreadLogBlocks" type="System.Int32" default="60"␣
↪comment="in seconds. Amount of time before cleanup runs."></entry>
        <entry key="heartbeatMessageInLog" type="System.Boolean" default="false"></
↪entry>
        <entry key="garbageCollectionMessageInLog" type="System.Boolean" default=
↪"false"></entry>
        <entry key="disableStacktrace" type="System.Boolean" default="false"></
↪entry>
        <entry key="useMailHashing" type="System.Boolean" default="true"></entry>
        <entry key="mailhashDir" type="System.String" default=".\mailHash"></entry>
        <entry key="mailhashIntervals" type="System.String" default="5,10,15,20,30,
↪35,60" comment="comma separated list of minutes between identical mails"></entry>
      </section>
```

16.10 Section “Main Mode

16.10.1 parallelWFExecutors

```
        <entry key="parallelWFExecutors" type="System.Int32" default="5"></entry>
```

This setting specifies the number of workflows that can be executed in parallel using sub processes of *kenaflow*.

16.10.2 rerHostEnabled

```
        <entry key="rerHostEnabled" type="System.Boolean" default="true"></entry>
```

This setting enables (by default) the “Remote Event Receiver” mode for this instance of _kenaflow.

This means that a sub process of *kenaflow* will listen for SharePoint and other remote events.

16.10.3 rerHostOnlyMode

```
        <entry key="rerHostOnlyMode" type="System.Boolean" default="false"></entry>
```

If enabled this *kenaflow* instance will ONLY execute the “Remove Event Receiver” sub process. No workflows will be executed.

This could be usefull to run the remove event receiver on a seperate machine that receives events for other *kenaflow* instances on other server.

Remote events are written to a disk folder or network share as configured in section *“RER”* of the global configuration.

16.10.4 slots

```
      <entry key="slots" type="System.String" comment="semicolon separated list␣
↪of integer values" default=""></entry>
```

This setting can be used to configure "execution slots".

Please read article *Execution Slots* of section "Advanced Configuration" for details.

16.10.5 disablePriorityAlgorithm

```
      <entry key="disablePriorityAlgorithm" type="System.Boolean" comment=""␣
↪default="false"></entry>
```

This setting can be used to switch off the priority engine for workflows.

Please read article *Workflow Priority* of section "Advanced Configuration" for details.

16.10.6 FirstRunTBEAddPercentage

```
      <entry key="FirstRunTBEAddPercentage" type="System.Int32" default="10"></
↪entry>
```

This setting specifies a kind of delay for the first start of a workflow *after a server reboot* or "if the main mode process" starts. It uses the percentage configured here multiplied with the workflows "TBE" ("time between execution") **and** multiplied with a random value to delay the first start of a workflow. This prevents *kenaflow* to run all workflows asap after an engine restart. To disable this feature you can set it to 0 (zero). If this feature is enabled (setting greater than 0), it uses a minimum of 10 seconds to be multiplied with the random value if the TBE * `FirstRunTBEAddPercentage` is lower than 10 seconds.

The (pseudo code) formula is: `random_double_between_0_and_1 * max( 10, (TBE * (FirstRunTBEAddPercentage / 100)) )` (function "max" means: "take the number that is greater"! It's not an error.)

16.10.7 Full Section Default

```
    <section name="mainmode">
      <entry key="parallelWFExecutors" type="System.Int32" default="5"></entry>
      <entry key="rerHostEnabled" type="System.Boolean" default="true"></entry>
      <entry key="rerHostOnlyMode" type="System.Boolean" default="false"></entry>
      <entry key="slots" type="System.String" comment="semicolon separated list␣
↪of integer values" default=""></entry>
      <entry key="disablePriorityAlgorithm" type="System.Boolean" comment=""␣
↪default="false"></entry>
      <entry key="FirstRunTBEAddPercentage" type="System.Int32" default="10"></
↪entry>
    </section>
```

16.11 Section “SharePointConnection

16.11.1 connectionStartThrottling

```
        <entry key="connectionStartThrottling" type="System.Int32" default="50"></
↪entry>
```

Each SharePoint connection is throttled this amount of time.

16.11.2 throttlingRetryCount

```
        <entry key="throttlingRetryCount" type="System.Int32" default="20"></entry>
```

When a SharePoint client request failed for some reasons it will be executed again. Here the count of possible retries is configured.

16.11.3 throttlingInitialDelay

```
        <entry key="throttlingInitialDelay" type="System.Int32" default="100"></
↪entry>
```

If the SharePoint customer request fails, it is executed again. The next executions are delayed. Each one more and more. The delay starts with the number of milliseconds configured here.

16.11.4 throttlingDelayFactor

```
        <entry key="throttlingDelayFactor" type="System.Double" default="1.2"></
↪entry>
```

Starting from the start delay, configured in the ‘throttlingInitialDelay’ setting, this delay is multiplied each time by the factor configured here, so that the time interval between two attempts always increases.

16.11.5 Full Section Default

```
      <section name="sharepointConnection">
        <entry key="connectionStartThrottling" type="System.Int32" default="50"></
↪entry>
        <entry key="throttlingRetryCount" type="System.Int32" default="20"></entry>
        <entry key="throttlingInitialDelay" type="System.Int32" default="100"></
↪entry>
        <entry key="throttlingDelayFactor" type="System.Double" default="1.2"></
↪entry>
      </section>
```

16.12 Section "MultiProcessing

16.12.1 testClientRunners

```
        <entry key="testClientRunners" type="System.Int32" default="1"></entry>
```

This is used for testing the multiprocessing architecture of *kenaflow*.

`kenaflow.exe --testmain` starts *this* amount of clients to simulate multi-processing.

16.12.2 retryCount

```
        <entry key="retryCount" type="System.Int32" default="10"></entry>
```

kenaflow executes various tasks in subprocesses. These subprocesses are monitored by the main process. If the contact could not be established more than this configured amount of times, then the subprocess is considered as "crashed" and the task is given to another sub process.

16.12.3 waitBeforeRetry

```
        <entry key="waitBeforeRetry" type="System.Int32" default="1000" comment=
↪"in milliseconds"></entry>
```

At least *this* amount of milliseconds must pass between two contacts to the sub process.

16.12.4 QueueIdleWait

```
        <entry key="QueueIdleWait" type="System.Int32" default="5000" comment="in␣
↪milliseconds"></entry>
```

If the workflow execution queue is empty the *kenaflow* mail process will wait *this* amount of time before it re-checks the queue.

16.12.5 parentProcessCheckInterval

```
        <entry key="parentProcessCheckInterval" type="System.Int32" default="1000"␣
↪comment="in milliseconds"></entry>
```

Each *kenaflow* sub process that was stated by the main process will monitor the main process. If this is dead the client will terminate itself.

16.12.6 heartbeatInterval

```
        <entry key="heartbeatInterval" type="System.Int32" default="1000" comment=
↪"in milliseconds"></entry>
```

Each *kenaflow* sub process that was stated by the main process sends an heartbeat to the main process.

16.12.7 heartbeatTimeout

```
        <entry key="heartbeatTimeout" type="System.Int32" default="60" comment="in␣
↪seconds"></entry>
```

The main process terminates a sub process if the last received heartbeat is older than *this* amount of seconds.

16.12.8 Full Section Default

```
      <section name="multiprocessing">
        <entry key="testClientRunners" type="System.Int32" default="1"></entry>
        <entry key="retryCount" type="System.Int32" default="10"></entry>
        <entry key="waitBeforeRetry" type="System.Int32" default="1000" comment=
↪"in milliseconds"></entry>
        <entry key="QueueIdleWait" type="System.Int32" default="5000" comment="in␣
↪milliseconds"></entry>
        <entry key="parentProcessCheckInterval" type="System.Int32" default="1000"␣
↪comment="in milliseconds"></entry>
        <entry key="heartbeatInterval" type="System.Int32" default="1000" comment=
↪"in milliseconds"></entry>
        <entry key="heartbeatTimeout" type="System.Int32" default="60" comment="in␣
↪seconds"></entry>
      </section>
```

16.13 Section “Lock

16.13.1 lockDir

```
        <entry key="lockDir" type="System.String" default=".\locks"></entry>
```

Workflows, lists and list items must be locked when they are processed.

The lock is released via temporary files in this directory.

For relative path specifications, the “ProgramDataFolder” is used as root (see setting in section *“Basic”*).

16.13.2 skippedItemCacheDir

```
        <entry key="skippedItemCacheDir" type="System.String" default=".\skipcache
↪"></entry>
```

! Currently not used.

16.13.3 maxWaitForLock_Item

```
        <entry key="maxWaitForLock_Item" type="System.Int32" default="5000"␣
↪comment="in millisecods"></entry>
```

Specifies how long to wait for the availability of a SharePoint list item when it is locked. (In milliseconds)

16.13.4 maxWaitForLock_List

```
        <entry key="maxWaitForLock_List" type="System.Int32" default="30000"␣
↪comment="in millisecods"></entry>
```

Specifies how long the availability of a SharePoint list should be waited for when it is locked. (In milliseconds)

16.13.5 maxWaitForLock_Workflow

```
        <entry key="maxWaitForLock_Workflow" type="System.Int32" default="60000"␣
↪comment="in millisecods"></entry>
```

Specifies how long the availability of a SharePoint list should be waited for when it is locked. (In milliseconds)

16.13.6 Full Section Default

```
      <section name="lock">
        <entry key="lockDir" type="System.String" default=".\locks"></entry>
        <entry key="skippedItemCacheDir" type="System.String" default=".\skipcache
↪"></entry>
        <entry key="maxWaitForLock_Item" type="System.Int32" default="5000"␣
↪comment="in millisecods"></entry>
        <entry key="maxWaitForLock_List" type="System.Int32" default="30000"␣
↪comment="in millisecods"></entry>
        <entry key="maxWaitForLock_Workflow" type="System.Int32" default="60000"␣
↪comment="in millisecods"></entry>
      </section>
```

16.14 Section “Mail

!!! We have a typo in the “SMTP” related config keys: We have named them “smpt”. We are sorry about that, but currently we cannot change this. Our plan is to fix thix during an update. We inform you here and in the *change log*

!! For anonymous SMTP server access please set `and `true` and `smptusername` to` (empty) !! !! For authentication to SMTP server using the credentials of the service account of kenaflow AND during debug using the credentials of the workflow developer please set `and `true` and `smptusername` to` (empty)

16.14.1 useSharePointForSend

```
        <entry key="useSharePointForSend" type="System.Boolean" default="false"␣
↪comment="if you client cannot send using SMTP than try this."></entry>
```

If your *kenaflow* server machine is not able to send mails directly by using SMTP you can leverage SharePoint to send mails.

This does **not** allow you to send attachments!

! This only works for SharePoint related workflows!

16.14.2 smptserver

```
        <entry key="smptserver" type="System.String"></entry>
```

The host name or IP of your SMTP server.

16.14.3 smptserverport

```
        <entry key="smptserverport" type="System.Int32" default="587"></entry>
```

The SMTP servers port.

16.14.4 smptusername

```
        <entry key="smptusername" type="System.String"></entry>
```

For SMTP authentication you can specify the username here.

16.14.5 smptpassword

```
        <entry key="smptpassword" type="System.String"></entry>
```

For SMTP authentication you can specify the password here.

16.14.6 smptUseSsl

```
        <entry key="smptUseSsl" type="System.Boolean" default="true"></entry>
```

Specifies whether the SMTP server is to be addressed using SSL. This is the default.

16.14.7 smptUseDefaultCredentials

```
        <entry key="smptUseDefaultCredentials" type="System.Boolean" default="true
↪" comment="used only if *smptusername* is empty!"></entry>
```

Use this setting to authenticate to SMTP server using the current Windows credentials.

During debug the credentials of the workflow developer are used.

During automatic run the credentials of the workflow engines service account is used.

16.14.8 mailSender

```
        <entry key="mailSender" type="System.String"></entry>
```

This email address is used as the senders address.

16.14.9 mailReplyto

```
        <entry key="mailReplyto" type="System.String"></entry>
```

This email address is used as reply-to address.

16.14.10 subjectPrefix

```
        <entry key="subjectPrefix" type="System.String" default=""></entry>
```

kenaflow can add an custom prefix to every send mails subject.

16.14.11 subjectSuffix

```
        <entry key="subjectSuffix" type="System.String" default=""></entry>
```

kenaflow can add an custom suffix to every send mails subject.

16.14.12 Full Section Default

```
      <section name="mail">
        <entry key="useSharePointForSend" type="System.Boolean" default="false"␣
↪comment="if you client cannot send using SMTP than try this."></entry>
        <entry key="smptserver" type="System.String"></entry>
        <entry key="smptserverport" type="System.Int32" default="587"></entry>
        <entry key="smptusername" type="System.String"></entry>
        <entry key="smptpassword" type="System.String"></entry>
        <entry key="smptUseSsl" type="System.Boolean" default="true"></entry>
        <entry key="smptUseDefaultCredentials" type="System.Boolean" default="true
↪" comment="used only if *smptusername* is empty!"></entry>
        <entry key="mailSender" type="System.String"></entry>
        <entry key="mailReplyto" type="System.String"></entry>
        <entry key="subjectPrefix" type="System.String" default=""></entry>
        <entry key="subjectSuffix" type="System.String" default=""></entry>
      </section>
```

16.15 Section “MailDev

The settings in this section affect all workflows that do not have their own setting for the following values in the workflow configuration file _wfconfig.ps1.

16.15.1 writeMailsToDiskDuringDebugDefaultSetting

```
        <entry key="writeMailsToDiskDuringDebugDefaultSetting" type="System.Boolean
↪" default="true"></entry>
```

Specifies that in the debug mode of workflow scripts the mails are not sent but written to text files.

This setting is used as default for each workflow that has *no* own setting for this property.

16.15.2 alwaysWriteMailsToDiskInsteadSending

```
      <entry key="alwaysWriteMailsToDiskInsteadSending" type="System.Boolean"␣
↪default="false"></entry>
```

Specifies that emails are **ALWAYS** written to text files instead of sending them.

This setting is used as default for each workflow that has *no* own setting for this property.

16.15.3 mailsOnDisk

```
      <entry key="mailsOnDisk" type="System.String" default=".\mails"></entry>
```

If mails are written to disk instead of sending them using SMTP they ware stored in this location.

The path is relative to the data directory of the *kenaflow* instance. You can also specify an absolute path to a custom location. The *kenaflow* service account need to have write permissions on the path.

16.15.4 Full Section Default

```
    <section name="maildev">
      <entry key="writeMailsToDiskDuringDebugDefaultSetting" type="System.Boolean
↪" default="true"></entry>
      <entry key="alwaysWriteMailsToDiskInsteadSending" type="System.Boolean"␣
↪default="false"></entry>
      <entry key="mailsOnDisk" type="System.String" default=".\mails"></entry>
    </section>
```

16.16 Section “RER

kenaflow can receive “remote event” sent by SharePoint or other messages send by third party applications by using an web API.

To enable this feature you need to configure this section.

!! When changing the Remote Event handler settings in the you maybe need to run `kenaflow.exe --updatenetsecurity` to grant access to the new URL.

16.16.1 rerRunnerSP2013

```
      <entry key="rerRunnerSP2013" type="System.Int32" default="1"></entry>
```

Here you can specify how many processes are used to process SharePoint 2013 related remote events. We call this “runner process”. If you have SharePoint 2013 workflows that are configured to receive events you need to set this setting at least to “1”.

!!! In DEVELOPER edition and FREE edition you cannot configure this setting. You will have always “1” runner process.

16.16.2 rerRunnerSP2016

```
        <entry key="rerRunnerSP2016" type="System.Int32" default="1"></entry>
```

Here you can specify how many processes are used to process SharePoint 2016 related remote events. We call this "runner process". If you have SharePoint 2016 workflows that are configured to receive events you need to set this setting at least to "1".

!!! In DEVELOPER edition and FREE edition you cannot configure this setting. You will have always "1" runner process.

16.16.3 rerRunnerSP2019

```
        <entry key="rerRunnerSP2019" type="System.Int32" default="1"></entry>
```

Here you can specify how many processes are used to process SharePoint 2019 related remote events. We call this "runner process". If you have SharePoint 2019 workflows that are configured to receive events you need to set this setting at least to "1".

!!! In DEVELOPER edition and FREE edition you cannot configure this setting. You will have always "1" runner process.

16.16.4 rerRunnerSPO

```
        <entry key="rerRunnerSPO" type="System.Int32" default="1"></entry>
```

Here you can specify how many processes are used to process SharePoint Online related remote events. We call this "runner process". If you have SharePoint Online workflows that are configured to receive events you need to set this setting at least to "1".

!!! In DEVELOPER edition and FREE edition you cannot configure this setting. You will have always "1" runner process.

16.16.5 rerRunnerPOSH

```
        <entry key="rerRunnerPOSH" type="System.Int32" default="1"></entry>
```

Here you can specify how many processes are used to trigger PowerShell based workflows that are not related to SharePoint. We call this "runner process". If you have "pure" PowerShell workflows that are configured to receive data you need to set this setting at least to "1".

!!! In DEVELOPER edition and FREE edition you cannot configure this setting. You will have always "1" runner process.

16.16.6 RemoteEventHandlerName

```
        <entry key="RemoteEventHandlerName" type="System.String" default="kenaflow
↪"></entry>
```

When a *kenflow* workflow is configured to receive events from SharePoint the URL of *kenaflow* is registered in SharePoint as "Remote Event Receiver". This registration requires a name. The name is build from this setting used as prefix and additionally the workflow ID to be unique.

16.16.7 rerQueueDir

```
        <entry key="rerQueueDir" type="System.String" default=".\queue"></entry>
```

Received events and data are stored in (encrypted) files on this before processing them using the "runner" processes of *kenaflow*.

Here you specify the location where the events are stored. You can specify a path relative to the `ProgramDataFolder` (default is .\queue) or a custom absolute path. *kenaflow* (its service account) need to have write permission at the configured location.

16.16.8 rerHost

```
        <entry key="rerHost" type="System.String" comment="dns of kenaflow"␣
↪default="+"></entry>
```

This setting is used for "Remote Event Receiver" URLs registered with workflows, e.g. SharePoint workflows that can be triggered by SharePoint.

You will need this setting to trigger workflows with custom data or unstructured data using the *kenaflow* web API.

Specify the FQDN or hostname of the kenaflow machine.

Use + for IP address based remote event receivers.

16.16.9 rerHostIp

```
        <entry key="rerHostIp" type="System.String" comment="ip of the server used␣
↪for remote events. Only required if 'rerHost' is '+'"></entry>
```

IP of the server to be used for remote events.

For generating an HTTP endpoint a host name (`rerHost`) or an IP (`rerHostIp`) is required.

This setting is only required if 'rerHost' is '+'.

16.16.10 rerHostUsesSSL

```
        <entry key="rerHostUsesSSL" type="System.Boolean" default="false"></entry>
```

kenaflow can receive data from SharePoint or by using the *kenaflow* web API using SSL. If you want to use SSL you must set this setting to `true`. You need to specify an SSL certificate using program switch *`--confighttp`*.

16.16.11 rerPort

```
        <entry key="rerPort" type="System.Int32" default="80"></entry>
```

kenaflow can receive data from SharePoint or by using the *kenaflow* web API using SSL. Here you specify the HTTP port. If you want to use SSL you need to specify the appropriate port (normally 443) yourself!

16.16.12 excludeCurrentUser

```
        <entry key="excludeCurrentUser" type="System.Boolean" default="true"␣
↪comment="Remote events related to the current user will be excluded to prevent␣
↪event floods."></entry>
```

As part of its normal function *kenaflow* changes SharePoint list items. If remote events are registered with a *kenaflow* workflow every change in a SharePoint list are send as remote events to *kenaflow*. Without this setting you would produce an endless loop: "User change in SharePoint" => trigger workflow by SharePoint remote event => *kenaflow* workflow changes the list item during processing => the change triggers the workflow by SharePoint remote event => *kenaflow* workflow changes the list item during processing the change triggers the workflow by SharePoint remote event => ... and so on ...

SharePoint stores the data-changing user in the remote event.

kenaflow reads this information and ignores remote events coming from the current user that runs *kenaflow*. This is the service account of *kenaflow*.

! You should not change this setting.

16.16.13 excludeUsers

```
        <entry key="excludeUsers" type="System.String" default="|app@sharepoint"␣
↪comment="exclude additional users. separate them by using ';'. all parts␣
↪(separated by ;) will be treated as regular expressions!"></entry>
```

Please read the description of setting `excludeCurrentUser` first.

Here you configure *patterns* for users that also should be ignored additionally to the "current user" (*kenaflow* service account).

First: you can specify multiple patterns by using ";" as separator.

Event triggering users are compared to each pattern configured here: If the event triggering user END WITH one of this patterns the event is ignored!

Default: "|app@sharepoint" => This is used to ignore ALL events triggered by "app credentials".

16.16.14 rerCheckDuringWorkflowExec

```
        <entry key="rerCheckDuringWorkflowExec" type="System.Boolean" default="true
↪"></entry>
```

kenaflow workflows are able to receive remote events from SharePoint. This requires a registration of the *kenaflow* workflow on each SharePoint list that is connected to a *kenaflow* workflow.

This registration can be forced using program switch *--registerrer*.

For removing the registration of a certain workflow you can use *--unregisterrer*.

if this setting is set to `true` (this is the default) on each workflow run the RER registration is checked. The registration on SharePoint added if missing and removed if present but not configured for the workflow. So you do not need to use `--registerrer` or `--unregisterrer` (when you want to disable remote events).

16.16.15 postponeEventOnScriptError

```
        <entry key="postponeEventOnScriptError" type="System.Int32" default="30"␣
↪comment="In Seconds"></entry>
```

In case of errors during processing an remote event this event will be postponed for the amout of seconds specified here.

16.16.16 encryptRemoteEvents

```
        <entry key="encryptRemoteEvents" type="System.Boolean" default="true"></
↪entry>
```

Remote events received from SharePoint or third party systems using the *kenaflow* web API are queued on disk. Here you specify whether *kenaflow* should or should not encrypt the queue files.

Default: true = encrypt

16.16.17 sslCertThumbprint

```
        <entry key="sslCertThumbprint" type="System.String" default=""></entry>
```

Using command line switch `--confighttp` you can register an SSL certificate for remote event endpoints (*Remote Event*).

If you register an SSL certificate it's thumbprint is written to this setting.

Please use command line switch `--securing` to grant access for users to the certificate.

16.16.18 rerReponseCheckEnabled

```
        <entry key="rerReponseCheckEnabled" type="System.Boolean" default="true"></
↪entry>
```

The main mode of kenaflow (`--run`) by default triggers the RER endpoints of *kenaflow* to check their availability. If the remote event receiver process (`kenaflow.exe --rer`) does not serve the endpoint it gets restarted.

This setting can be used to disable this check mechanism.

The following settings can be used to fine tune the RER checker (if enabled).

16.16.19 rerReponseCheckInterval

```
        <entry key="rerReponseCheckInterval" type="System.Int32" comment="in␣
↪milliseconds" default="5000"></entry>
```

This sets the check interval.

16.16.20 rerReponseCheckTimeout

```
        <entry key="rerReponseCheckTimeout" type="System.Int32" comment="in␣
↪milliseconds" default="5000"></entry>
```

This sets the timeout for HTTP endpoint check.

16.16.21 rerReponseCheckFailCount

```
        <entry key="rerReponseCheckFailCount" type="System.Int32" default="5"></
↪entry>
```

After this amount of failed checks the RER process gets terminated and restarted.

16.16.22 rerReponseCheckEnabledSPRER

```
        <entry key="rerReponseCheckEnabledSPRER" type="System.Boolean" default=
↪"true"></entry>
```

This enables or disables the SharePoint remote event receiver endpoint checker.

16.16.23 rerReponseCheckEnabledLINK

```
        <entry key="rerReponseCheckEnabledLINK" type="System.Boolean" default="true
↪"></entry>
```

This enables or disables the link remote event receiver endpoint checker.

16.16.24 rerReponseCheckEnabledCDA

```
        <entry key="rerReponseCheckEnabledCDA" type="System.Boolean" default="true
↪"></entry>
```

This enables or disables the custom data remote event receiver endpoint checker.

16.16.25 rerReponseCheckEnabledUDA

```
        <entry key="rerReponseCheckEnabledUDA" type="System.Boolean" default="true
↪"></entry>
```

This enables or disables the unstructured data remote event receiver endpoint checker.

16.16.26 Full Section Default

```
      <section name="rer">
        <entry key="rerRunnerSP2013" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerSP2016" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerSP2019" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerSPO" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerPOSH" type="System.Int32" default="1"></entry>
        <entry key="RemoteEventHandlerName" type="System.String" default="kenaflow
↪"></entry>
```

(continues on next page)

(continued from previous page)

```
        <entry key="rerQueueDir" type="System.String" default=".\queue"></entry>
        <entry key="rerHost" type="System.String" comment="dns of kenaflow"␣
↪default="+"></entry>
        <entry key="rerHostIp" type="System.String" comment="ip of the server used␣
↪for remote events. Only required if 'rerHost' is '+'"></entry>
        <entry key="rerHostUsesSSL" type="System.Boolean" default="false"></entry>
        <entry key="rerPort" type="System.Int32" default="80"></entry>
        <entry key="excludeCurrentUser" type="System.Boolean" default="true"␣
↪comment="Remote events related to the current user will be excluded to prevent␣
↪event floods."></entry>
        <entry key="excludeUsers" type="System.String" default="|app@sharepoint"␣
↪comment="exclude additional users. separate them by using ';'. all parts␣
↪(separated by ;) will be treated as regular expressions!"></entry>
        <entry key="rerCheckDuringWorkflowExec" type="System.Boolean" default="true
↪"></entry>
        <entry key="postponeEventOnScriptError" type="System.Int32" default="30"␣
↪comment="In Seconds"></entry>
        <entry key="encryptRemoteEvents" type="System.Boolean" default="true"></
↪entry>
        <entry key="sslCertThumbprint" type="System.String" default=""></entry>
        <entry key="rerReponseCheckEnabled" type="System.Boolean" default="true"></
↪entry>
        <entry key="rerReponseCheckInterval" type="System.Int32" comment="in␣
↪milliseconds" default="5000"></entry>
        <entry key="rerReponseCheckTimeout" type="System.Int32" comment="in␣
↪milliseconds" default="5000"></entry>
        <entry key="rerReponseCheckFailCount" type="System.Int32" default="5"></
↪entry>
        <entry key="rerReponseCheckEnabledSPRER" type="System.Boolean" default=
↪"true"></entry>
        <entry key="rerReponseCheckEnabledLINK" type="System.Boolean" default="true
↪"></entry>
        <entry key="rerReponseCheckEnabledCDA" type="System.Boolean" default="true
↪"></entry>
        <entry key="rerReponseCheckEnabledUDA" type="System.Boolean" default="true
↪"></entry>
    </section>
```

16.17 Section “cmdletConfic

In this configuration sections there are stored some settings for certain *kenaflow* Cmdlets.

16.17.1 Settings for Cmdlet `Invoke-KFFileTemplate`

16.17.1.1 `FileTemplateStartMarker`

```
        <entry key="FileTemplateStartMarker" type="System.String" comment="For␣
↪cmdlet Invoke-KFFileTemplate" default="%{"></entry>
```

kenaflow provides the Cmdlet `Invoke-KFFileTemplate` that is able to replace placeholders in WORD and POWERPOINT files using the kenaflow *template engine*.

To be able to use the template engine you need to mark the placeholders additionally with the prefix specified here. The suffix is specified in the setting `FileTemplateEndMarker` (description below).

Example:

```
    %{{{Title}}}%
```

`{{Title}}` is processed by the _kenaflow template engine. `%{` and `}%` are the markers for the placeholder.

16.17.1.2 FileTemplateEndMarker

```
        <entry key="FileTemplateEndMarker" type="System.String" comment="For␣
↪cmdlet Invoke-KFFileTemplate" default="}%"></entry>
```

Please read the description of setting `FileTemplateStartMarker` above.

16.17.2 Full Section Default

```
      <section name="cmdletConfig">
        <entry key="FileTemplateStartMarker" type="System.String" comment="For␣
↪cmdlet Invoke-KFFileTemplate" default="%{"></entry>
        <entry key="FileTemplateEndMarker" type="System.String" comment="For␣
↪cmdlet Invoke-KFFileTemplate" default="}%"></entry>
      </section>
```

16.18 Section "Links

16.18.1 linkClickAnswer

```
        <entry key="linkClickAnswer" type="System.String" default=""> <![CDATA[ ...
↪ ]]> </entry>
```

kenaflow can provide "rich links" that trigger actions in workflows by sending data.

This is described here: *Links*.

There you specify the template for the website that is shown when a user opens the link url in the browser.

BUT if the link is expired the template specified in setting `linkClickExpired` is shown.

The default for the setting can be found in the "Full Section Default" below.

16.18.2 linkClickExpired

```
        <entry key="linkClickExpired" type="System.String" default=""> <![CDATA[ ..
↪. ]]> </entry>
```

kenaflow can provide "rich links" that trigger actions in workflows by sending data.

This is described here: *Links*.

If the user opens the link in the browser and the link is expired the website specified there is shown.

The default for the setting can be found in the "Full Section Default" below.

16.18.3 Full Section Default

```
      <section name="links">
        <entry key="linkClickAnswer" type="System.String" default="">
          <![CDATA[
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>kenaflow</title>
    <style>
    .loginDiv {
        position: absolute;
        left: 50%;
        top: 50%;
        text-align: center;
        background-color: #108f44;
        width:546px;
        height:265px;
        margin-left: -273px; /*half width*/
        margin-top: -132px; /*half height*/
    }
    /* latin-ext */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/RlBXAIuiO5GvH9-0-JbBlw.woff2) format('woff2');
      unicode-range: U+0100-024F, U+1E00-1EFF, U+20A0-20AB, U+20AD-20CF, U+2C60-
↪2C7F, U+A720-A7FF;
    }
    /* latin */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/4hYyXH_8WmbBLamf6WjLwg.woff2) format('woff2');
      unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02C6, U+02DA, U+02DC,␣
↪U+2000-206F, U+2074, U+20AC, U+2212, U+2215;
    }
    p { font-size:40px;font-family:'Dosis';color:white}
    </style>
    </head>
    <div class="loginDiv">
    <p>We got your answer!<br/>Thank you.</p>
    <p><img src="
↪iVBORw0KGgoAAAANSUhEUgAAAMgAAABACAIAAADQ2a98AAABgWlDQ1BzUkdCIElFQzYxOTY2LTIuMQAAKJF1kc8rRFEUxz8
↪4CtspaKSIlZWdNbNBz3oyaSebczj2f+733nO49F6yhlJLWa9yQzuS0gN/
↪rnF9YdNY900AnrVixhxVdnQlOhKhqH3dYzHjTb9aqfu5fa4zGdAUs9cJjiqrlhCeFp9dyqsnbwu1KMhwVPhXu0+SCwremHi
↪HhxMsU4PoYZZFTmYfrxMCArquS7i/mzZCVXkVklj8YKCZLk6BN1VarHJMZFj8lIkTf7/
↪7evenzIU6ru8ELtk2G89UDdFnwXDOPz0DC+j8D2CBeZcn72AEbeRS+UNdc+NG/
↪A2WVZi+zA+SZ0PKhhLVyUbOLWeBxeT6BpAdquwb5U6tnvPsf3EFqXr7qC3T3olfPNyz8ajGfD2mVUoQAAAAlwSFlzAAAuIwA
↪dgAADx1JREFUeJztnWlYU8cax+9/4AxgUsiRHgOgUFrbo9dyqsnbwu1KMhwVPhXu0+SCwremHi
↪zNn5p0JUNS/
↪8wMMRt5Qp9oBzF8TLCwMKWBhYUgBCwtDClhYGFLAwsKQAhYWhhSwsDCkQBtX6f12wUEOgUTy1P1foxoy5O0S5q/
↪A+ISlqaxhbmBFJNXoKvL2B/
↪MXAX8KMaSAhYUhBSwsDClgYWFIAQsLQwpYWBhSwMLCkAIWFoYUsLAwpPAqCosKEKbJClQ3nmpHMBPn1RKWhSJjl6Xv/
↪VXf//Lmz3baZlPtzhSjTVWYRnm1HtDYGd9aIUkoAmW5jkW47VI/
↪xwAlGn3C7SxS0T4YuIdIbk74V66oSx4OTipUgC0z3F53jbA2su3t7ytqyC9oKkxtyKRRFHYu/
↪pAoFhL7T55kYAr9HJ0pFpY9XWWVmVf47LCZunLonxgKirbGs4ik0p/zdT/
↪qsvYNj7eQTVNQnG+5cL7lwu7rTyrb62SvjkqhTJGDY2JqhMWgUFbr24XZBSy296P+OR8/
↪SQRrTCdUJUtuawlTedrk+zNhJltY85TVV1j4hDmHGWlNn+RT/
↪yn4m30YYUukkuTCG21drXZGs1IEjaG61lPo2HiZJGFpUBRWGTmG2wd52HhNzhn/
↪pJjqDQ0Jfn9w4d0Hp6bQmZeBdGF5MnVWWC4OcQ7TUdMbe62m9kZOZ6sc3bBRUvbWGxqgZPEqsno6kU0FmKOsZq/
↪CAIBAQABAoIBAQCJjLEmrO5W7kC/bAu2StO89OyQ/UjIjefXA4C5ImO+5kxbHXMVpWlFrRV3uGU1/
↪b3DTq1BUfDSNDFhGpioGzDpzObOtubuNja3vELcM9JPlq4pYec3l07gSlk0pdlqRlZapiwNo6bO1sr2upwOTrVYRBRYrWOp
↪LB011zz+SiZUZl2NikH2fZ0lX97vL/YfumwKtzO1q+uHTrZnE/k2DINvl55ANnxWbEp6ce/
↪c9uwct5q2Vr13Np34j4hJGtCU3rTzOsNtzWGmsNjcl2izjP3fokqS3o0IEY5XkxdZarSNJlNuToqmgFqxgDQLu7O7Ol44WU
↪ENKvPmL+8CqCOdefS1w2hA3PFEEH9v7ejOvF9Y/maG4Va+wbND1adpjLFKPbc2ueRWct39JAFHIm9/
```

(continues on next page)

(continued from previous page)

```
    </div>
    <body>
    </body>
    </html>
          ]]>
        </entry>
        <entry key="linkClickExpired" type="System.String" default="">
          <![CDATA[
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>kenaflow</title>
    <style>
    .loginDiv {
        position: absolute;
        left: 50%;
        top: 50%;
        text-align: center;
        background-color: #108f44;
        width:546px;
        height:265px;
        margin-left: -273px; /*half width*/
        margin-top: -132px; /*half height*/
    }
    /* latin-ext */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/RlBXAIuiO5GvH9-0-JbBlw.woff2) format('woff2');
      unicode-range: U+0100-024F, U+1E00-1EFF, U+20A0-20AB, U+20AD-20CF, U+2C60-
↪2C7F, U+A720-A7FF;
    }
    /* latin */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/4hYyXH_8WmbBLamf6WjLwg.woff2) format('woff2');
      unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02C6, U+02DA, U+02DC,␣
↪U+2000-206F, U+2074, U+20AC, U+2212, U+2215;
    }
    p { font-size:40px;font-family:'Dosis';color:white}
    </style>
    </head>
    <div class="loginDiv">
    <p>The link has expired!<br/>Sorry.</p>
    <p><img src="
```

```
↪iVBORw0KGgoAAAANSUhEUgAAAMgAAABACAIAAADQ2a98AAABgWlDQ1BzUkdCIElFQzYxOTY2LTIuMQAAKJF1kc8rRFEUxz8
↪4CtspaKSIlZWdNbNBz3oyaSebczj2f+733nO49F6yhlJLWa9yQzuS0gN/
↪rnF9YdNY900AnrVixhxVdnQlOhKhqH3dYzHjTb9aqfu5fa4zGdAUs9cJjiqrlhCeFp9dyqsnbwu1KMhwVPhXu0+SCwremHi
↪HhxMsU4PoYZZFTmYfrxMCArquS7i/mzZCVXkVklj8YKCZLk6BN1VarHJMZFj8lIkTf7/
↪7evenzIU6ru8ELtk2G89UDdFnwXDOPz0DC+j8D2CBeZcn72AEbeRS+UNdc+NG/
↪A2WVZi+zA+SZ0PKhhLVyUbOLWeBxeT6BpAdquwb5U6tnvPsf3EFqXr7qC3T3olfPNyz8ajGfD2mVUoQAAAAlwSFlzAAAuIw
↪dgAADx1JREFUeJztnWlYU8cax9+EQIKEfQcxsi+yKagoyKIg07ig7dXaVmvrrUtre2+1VXut12Jb69XW0udW66PXWhUrBUU
↪zNn5p0JUNS/
↪8wMMRt5Qp9oBzF8TLCwMKWBhYUgBCwtDClhYGFLAwsKQAhYWhhSwsDCkQBtX6f12wUEOgUTy1P1foxoy5O0S5q/
↪A+ISlqaxhbmBFJNXoKvL2B/
↪MXAX8KMaSAhYUhBSwsDClgYWFIAQsLQwpYWBhSwMLCkAIWFoYUsLAwpPAqCosKEKbJClQ3nmpHMBPn1RKWhSJjl6Xv/
↪VXf//Lmz3baZlPtzhSjTVWYRnm1HtDYGd9aIUkoAmW5jkW47VI/
↪xwAlGn3C7SxS0T4YuIdIbk74V66oSx4OTipUgC0z3F53jbA2su3t7ytqyC9oKkxtyKRRFHYu/
↪pAoFhL7T55kYAr9HJ0pFpY9XWWVmVf47LCZunLonxgKirbGs4ik0p/zdT/
↪[illegible]svYNj7eQTVNQnG+5cL7lwu7rTyrb62SvjkqhTJGDY2JqhMWgUFb[illegible]/
↪SQRrTCdUJUtuawlTcdrk+zNhJltY85TVV1j4hDmHGWlNn+RT/
↪yn4m30YYUukkuTCG21drXZGs1IEjaG61lPo2HiZJGFpUBRWGTmG2wd52HhNzhn/
↪pJjqDQ0Jfn9w4d0Hp6bQmZeBdGF5MnVWWC4OcQ7TUdMbe62m9kZOZ6sc3bBRUvbWGxqgZPEqsno6kU0FmKOsZq/
↪OstBiqTJUa3j1FYL6rO5W7kC/bAu2StO89OyQ/UjIjefXA4C5ImO+5kxbHXMVpWlFrRV3uGU1/
```


(continues on next page)

(continued from previous page)

```
    </div>
    <body>
    </body>
    </html>
          ]]>
        </entry>
      </section>
```

16.19 Section “DoNotTouch

Settings in this sections are not inteded to be changed by *kenaflow* administrators. They are used by the engine internally.

16.19.1 instanceId

```
        <entry key="instanceId" type="System.Guid" comment="do not change!!"␣
↪default="{00000000-0000-0000-0000-000000000000}"></entry>
```

ID of the *kenaflow* instance. It is set during installation and must not be changed.

16.19.2 configFileVersion

```
        <entry key="configFileVersion" type="System.String" comment="do not change␣
↪this!" default="4.0"/>
```

The version number of the config file.

16.19.3 productVersion

```
        <entry key="productVersion" type="System.String" comment="do not change␣
↪this!" default="4.0.0.0"/>
```

This setting contains the current installed version of *kenaflow*.

16.19.4 val1

```
        <entry key="val1" type="System.Boolean" default="false" comment="please␣
↪dont touch this"></entry>
```

Internally used.

16.19.5 val2

```
        <entry key="val2" type="System.Boolean" default="false" comment="please␣
↪dont touch this"></entry>
```

Internally used.

16.19.6 Full Section Default

```
      <section name="donttouchthissection">
        <entry key="instanceId" type="System.Guid" comment="do not change!!"␣
↪default="{00000000-0000-0000-0000-000000000000}"></entry>
        <entry key="configFileVersion" type="System.String" comment="do not change␣
↪this!" default="4.0"/>
        <entry key="productVersion" type="System.String" comment="do not change␣
↪this!" default="4.0.0.0"/>
        <entry key="val1" type="System.Boolean" default="false" comment="please␣
↪dont touch this"></entry>
        <entry key="val2" type="System.Boolean" default="false" comment="please␣
↪dont touch this"></entry>
      </section>
```

16.20 kenaflow.conf

Here you can find the complete content of the file `kenaflow.conf` after the setup.

```
    <?xml version="1.0" encoding="utf-8"?>
    <config>
      <section name="aftersetup">
        <entry key="adminMails" type="System.String" comment="split by ;"></entry>
        <entry key="WorkflowRoot" type="System.String" default=""></entry>
        <entry key="encryptionPassphrase" type="System.String" comment="do not␣
↪change after first usage." default="kenafl0W!"></entry>
        <entry key="activeDirectoryDomain" type="System.String" comment="please␣
↪specify your domain as FQDN" default="."></entry>
        <entry key="instanceName" type="System.String" comment="if not specified␣
↪its the instance id" default=""></entry>
      </section>
      <section name="licensing">
        <entry key="licensingMailAddress" type="System.String" default=""></entry>
        <entry key="Serial" type="System.String"></entry>
        <entry key="ActivationCode" type="System.String"></entry>
        <entry key="lastActivationCode" type="System.String" default=""></entry>
        <entry key="lastOffineDeactivationCode" type="System.String" default=""></
↪entry>
      </section>
      <section name="basic">
        <entry key="database" type="System.String" comment="SQLite database␣
↪location" default=".\kenaflow.sqlite"></entry>
        <entry key="globalFloodProtectionFile" type="System.String" default=".\
↪adminAlertFloodProtection.txt"></entry>
        <entry key="globalFloodProtectionTBE" type="System.Int32" default="300"␣
↪comment="in seconds"></entry>
        <entry key="globalWorkflowExecutionTBE" type="System.Int32" default="300"␣
↪comment="in seconds"></entry>
        <entry key="globalWorkflowStateTBE" type="System.Int32" default="60"␣
↪comment="in seconds"></entry>
        <entry key="ConfigEditor" type="System.String" default="notepad.exe"></
↪entry>
```

(continues on next page)

(continued from previous page)

```
        <entry key="EncryptConfig" type="System.Boolean" default="true"></entry>
        <entry key="Branch" type="System.String" default="release-4-0" comment=
↪"for updates..."></entry>
        <entry key="InfoServerPort" type="System.Int32" default="-1"></entry>
        <entry key="ProgramDataFolder" type="System.String" default=""></entry>
        <entry key="CustomLibrary" type="System.String" default="" comment=
↪"relative to program folder"></entry>
        <entry key="VSCodeDir" type="System.String" default=".\vscode" comment=
↪"please use the version of Visual Studio Code that we supply."></entry>
        <entry key="encryptSerializedWFCONFIG" type="System.Boolean" default="true
↪"></entry>
        <entry key="allowPowerShellPlaceholders" type="System.Boolean" default=
↪"false"></entry>
        <entry key="PluginsEnabled" type="System.Boolean" default="true"></entry>
        <entry key="PluginsFolder" type="System.String" default=".\Plugins"␣
↪comment="relative path to program folder or absolute path"></entry>
        <entry key="PluginsInfoDir" type="System.String" default=".\PluginInfo"␣
↪comment="relative path to program data folder or absolute path"></entry>
        <entry key="httpSecurityProtocolType" type="System.String" default="Tls12"␣
↪comment="Values:Tls12,Tls,Tls11,Ssl3 (sperated by comma)"></entry>
      </section>
      <section name="storage">
        <entry key="storageEnabled" type="System.Boolean" default="true"></entry>
        <entry key="storagePath" type="System.String" default=".\storage"></entry>
        <entry key="storageMaxSize" type="System.Int64" default="1073741824"␣
↪comment="1GB"></entry>
      </section>
      <section name="monitoring">
        <entry key="usePerformanceCounter" type="System.Boolean" default="false"></
↪entry>
        <entry key="WorkReportHookAddress" type="System.String" default=""></entry>
        <entry key="ClientStartReportHookAddress" type="System.String" default="">
↪</entry>
        <entry key="PingICMP" type="System.String" comment="[ip/host]|[time-
↪between]|[max-ok-latency-msec]|[stop-after-failed-count]|return-to-normal-after-
↪count]~..."></entry>
        <entry key="PingWebHook" type="System.String" comment="[url]|[time-
↪between]|[expected-http-status-code]|[stop-after-failed-count]|return-to-normal-
↪after-count]~..."></entry>
        <entry key="StopCountFailedPings" type="System.Int32" comment="how much␣
↪ping conditions must fail from setting above before pausing the engine?" default=
↪"1"></entry>
        <entry key="ErrorWebHook" type="System.String" comment="[url]|[level:␣
↪0=waring, 1=error, 2 = critical]|['get' or 'post']~..."></entry>
        <entry key="FreeDiskSpaceDrives" type="System.String" comment="example:
↪'C:;D:'. separated by ;" default=""></entry>
        <entry key="FreeDiskSpaceThreshold" type="System.Int32" comment="in␣
↪megabyte!" default="500"></entry>
        <entry key="FreeDiskSpaceWebHook" type="System.String" comment="[url]|~...
↪"></entry>
      </section>
      <section name="runtime">
        <entry key="ItemReCheckTime" type="System.Int32" default="60"></entry>
        <entry key="ItemBatchCount" type="System.Int32" default="1000"></entry>
        <entry key="MaxRuntimeOfChildProcess" type="System.Int32" default="86400"␣
↪comment="Time in Sec. before a child process is getting restarted - for security␣
↪reasons."></entry>
        <entry key="VariationOfMaxRuntimeOfChildProcess" type="System.Int32"␣
↪default="14400" comment="Time in Sec. before a child process is getting␣
↪restarted - for security reasons."></entry>
        <entry key="MaxExecutionTime" type="System.Int32" default="600"></entry>
```

(continues on next page)

(continued from previous page)

```
        <entry key="UserProfileCacheLifetime" type="System.Int32" default="60"␣
↪comment="in seconds"></entry>
        <entry key="defaultScriptRunTimeout" type="System.Int32" default="600"␣
↪comment="in seconds"></entry>
        <entry key="culture" type="System.Int32" default="1033" comment="This␣
↪culture overwrites the users culture. Set to -1 to disable."></entry>
        <entry key="poshQueueSize" type="System.Int32" default="5"></entry>
            <entry key="rerRunParallel" type="System.Boolean" default="false"></
↪entry>
            <entry key="rerRunParallelMaxThreads" type="System.Int32" default="5">
↪</entry>
      </section>
      <section name="logging">
        <entry key="useSerilog" type="System.Boolean" default="false"></entry>
        <entry key="serilogBaseTemplateConsole" type="System.String" default="[
↪{Level}] {Message}{NewLine}{Exception}"></entry>
        <entry key="serilogBaseTemplateFile" type="System.String" default="[{Level}
↪] {Message}{NewLine}{Exception}"></entry>
        <entry key="serilogTemplate" type="System.String" default="{TS:HH:mm:ss}
↪{Process} {Event} {ThreadId} {Correlation} {Text:l}"></entry>
        <entry key="minConsoleLogLevel" type="System.Int32" default="2" comment=
↪"0=debug, 1=output, 2=verbose, 3=warning, 4=normal, 5=error"></entry>
        <entry key="minFileLogLevel" type="System.Int32" default="0" comment=
↪"0=debug, 1=output, 2=verbose, 3=warning, 4=normal, 5=error"></entry>
        <entry key="dontLogIfMainWindowIsHidden" type="System.Boolean" default=
↪"true" comment="if the kenaflow process window is hidden no log info is written␣
↪to the console"></entry>
        <entry key="logDir" type="System.String" default=".\log"></entry>
        <entry key="logEnabled" type="System.Boolean" default="true"></entry>
        <entry key="logOnlyMainProcess" type="System.Boolean" default="true">
↪</entry>
        <entry key="logMemMaxAge" type="System.Int32" default="3600" comment=
↪"seconds"></entry>
        <entry key="logFileMaxAge" type="System.Int32" default="14" comment="days">
↪</entry>
        <entry key="logCleanupRunner" type="System.Int32" default="120" comment=
↪"in seconds. Amount of time before cleanup runs."></entry>
        <entry key="logfileCleanupRunner" type="System.Int32" default="3600"␣
↪comment="in seconds. Amount of time before log file cleanup runs."></entry>
        <entry key="logFCE" type="System.Boolean" default="false"></entry>
        <entry key="logOutput" type="System.Boolean" default="false"></entry>
        <entry key="sendOutputToMainProcess" type="System.Boolean" default="false">
↪</entry>
        <entry key="logfileLockTimeout" type="System.Int32" default="500" comment=
↪"in milliseconds"></entry>
        <entry key="logCleanupThreadLogBlocks" type="System.Int32" default="60"␣
↪comment="in seconds. Amount of time before cleanup runs."></entry>
        <entry key="heartbeatMessageInLog" type="System.Boolean" default="false"></
↪entry>
        <entry key="garbageCollectionMessageInLog" type="System.Boolean" default=
↪"false"></entry>
        <entry key="disableStacktrace" type="System.Boolean" default="false"></
↪entry>
        <entry key="exceptionAsJsonToConsole" type="System.Boolean" default="false
↪"></entry>
        <entry key="useMailHashing" type="System.Boolean" default="true"></entry>
        <entry key="mailhashDir" type="System.String" default=".\mailHash"></entry>
        <entry key="mailhashIntervals" type="System.String" default="5,10,15,20,30,
↪35,60" comment="comma separated list of minutes between identical mails"></entry>
      </section>
      <section name="mainmode">
```

(continues on next page)

(continued from previous page)

```
        <entry key="parallelWFExecutors" type="System.Int32" default="5"></entry>
        <entry key="rerHostEnabled" type="System.Boolean" default="true"></entry>
        <entry key="rerHostOnlyMode" type="System.Boolean" default="false"></entry>
        <entry key="slots" type="System.String" comment="semicolon separated list␣
↪of integer values" default=""></entry>
        <entry key="disablePriorityAlgorithm" type="System.Boolean" comment="␣
↪default="false"></entry>
        <entry key="FirstRunTBEAddPercentage" type="System.Int32" default="10"></
↪entry>
      </section>
      <section name="sharepointConnection">
        <entry key="connectionStartThrottling" type="System.Int32" default="50"></
↪entry>
        <entry key="throttlingRetryCount" type="System.Int32" default="20"></entry>
        <entry key="throttlingInitialDelay" type="System.Int32" default="100"></
↪entry>
        <entry key="throttlingDelayFactor" type="System.Double" default="1.2"></
↪entry>
        <entry key="connectionTimeout" type="System.Int32" default="10000"></entry>
      </section>
      <section name="multiprocessing">
        <entry key="testClientRunners" type="System.Int32" default="1"></entry>
        <entry key="retryCount" type="System.Int32" default="10"></entry>
        <entry key="waitBeforeRetry" type="System.Int32" default="1000" comment=
↪"in milliseconds"></entry>
        <entry key="QueueIdleWait" type="System.Int32" default="5000" comment="in␣
↪milliseconds"></entry>
        <entry key="parentProcessCheckInterval" type="System.Int32" default="1000"␣
↪comment="in milliseconds"></entry>
        <entry key="heartbeatInterval" type="System.Int32" default="1000" comment=
↪"in milliseconds"></entry>
        <entry key="heartbeatTimeout" type="System.Int32" default="60" comment="in␣
↪seconds"></entry>
      </section>
      <section name="lock">
        <entry key="lockDir" type="System.String" default=".\locks"></entry>
        <entry key="skippedItemCacheDir" type="System.String" default=".\skipcache
↪"></entry>
        <entry key="maxWaitForLock_Item" type="System.Int32" default="5000"␣
↪comment="in millisecods"></entry>
        <entry key="maxWaitForLock_List" type="System.Int32" default="30000"␣
↪comment="in millisecods"></entry>
        <entry key="maxWaitForLock_Workflow" type="System.Int32" default="60000"␣
↪comment="in millisecods"></entry>
      </section>
      <section name="mail">
        <entry key="useSharePointForSend" type="System.Boolean" default="false"␣
↪comment="if you client cannot send using SMTP than try this."></entry>
        <entry key="smptserver" type="System.String" default=""></entry>
        <entry key="smptserverport" type="System.Int32" default="587"></entry>
        <entry key="smptusername" type="System.String" default=""></entry>
        <entry key="smptpassword" type="System.String" default=""></entry>
        <entry key="smptUseSsl" type="System.Boolean" default="true"></entry>
        <entry key="smptUseDefaultCredentials" type="System.Boolean" default="true
↪" comment="used only if *smptusername* is empty!"></entry>
        <entry key="mailSender" type="System.String" default=""></entry>
        <entry key="mailReplyto" type="System.String" default=""></entry>
        <entry key="subjectPrefix" type="System.String" default=""></entry>
        <entry key="subjectSuffix" type="System.String" default=""></entry>
      </section>
      <section name="maildev">
```

(continues on next page)

(continued from previous page)

```
        <entry key="writeMailsToDiskDuringDebugDefaultSetting" type="System.Boolean
↪" default="true"></entry>
        <entry key="alwaysWriteMailsToDiskInsteadSending" type="System.Boolean"␣
↪default="false"></entry>
        <entry key="mailsOnDisk" type="System.String" default=".\mails"></entry>
      </section>
      <section name="rer">
        <entry key="useInstanceIdForSeparation" type="System.Boolean" default="true
↪"></entry>
        <entry key="rerRunnerSP2013" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerSP2016" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerSP2019" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerSPO" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerSPON" type="System.Int32" default="1"></entry>
        <entry key="rerRunnerPOSH" type="System.Int32" default="1"></entry>
        <entry key="RemoteEventHandlerName" type="System.String" default="kenaflow
↪"></entry>
        <entry key="rerQueueDir" type="System.String" default=".\queue"></entry>
        <entry key="rerPermQueueDir" type="System.String" default=".\permqueue"></
↪entry>
        <entry key="rerResponseQueueDir" type="System.String" default=".\
↪responsequeue"></entry>
        <entry key="keepProcessedEventsDir" type="System.String" comment="Leave␣
↪empty to remove processed events"></entry>
        <entry key="rerHost" type="System.String" comment="dns of kenaflow"␣
↪default="+"></entry>
        <entry key="rerHostIp" type="System.String" comment="ip of the server used␣
↪for remote events. Only used if 'rerHost' is '+'"></entry>
        <entry key="rerHostUsesSSL" type="System.Boolean" default="false"></entry>
        <entry key="rerPort" type="System.Int32" comment="default (if -1 or empty)␣
↪is 80 or 443 based on 'rerHostUseSSL'" default="-1"></entry>
        <entry key="excludeCurrentUser" type="System.Boolean" default="true"␣
↪comment="Remote events related to the current user will be excluded to prevent␣
↪event floods."></entry>
        <entry key="excludeUsers" type="System.String" default="|app@sharepoint"␣
↪comment="exclude additional users. separate them by using ';'. all parts␣
↪(separated by ;) will be treated as regular expressions!"></entry>
        <entry key="rerCheckDuringWorkflowExec" type="System.Boolean" default="true
↪"></entry>
        <entry key="postponeEventOnScriptError" type="System.Int32" default="5"␣
↪comment="In Seconds"></entry>
        <entry key="eventMaxAge" type="System.Int32" default="120" comment="In␣
↪Seconds"></entry>
        <entry key="encryptRemoteEvents" type="System.Boolean" default="true"></
↪entry>
        <entry key="sslCertThumbprint" type="System.String" default=""></entry>
        <entry key="rerReponseCheckEnabled" type="System.Boolean" default="true"></
↪entry>
        <entry key="rerReponseCheckInterval" type="System.Int32" comment="in␣
↪milliseconds" default="5000"></entry>
        <entry key="rerReponseCheckTimeout" type="System.Int32" comment="in␣
↪milliseconds" default="5000"></entry>
        <entry key="rerReponseCheckFailCount" type="System.Int32" default="5"></
↪entry>
        <entry key="rerReponseCheckEnabledSPRER" type="System.Boolean" default=
↪"true"></entry>
        <entry key="rerReponseCheckEnabledLINK" type="System.Boolean" default="true
↪"></entry>
        <entry key="rerReponseCheckEnabledCDA" type="System.Boolean" default="true
↪"></entry>
        <entry key="rerReponseCheckEnabledUDA" type="System.Boolean" default="true
↪"></entry>
```

(continues on next page)

(continued from previous page)

```
      </section>
      <section name="cmdletConfig">
        <entry key="FileTemplateStartMarker" type="System.String" comment="For␣
↪cmdlet Invoke-KFFileTemplate" default="%{"></entry>
        <entry key="FileTemplateEndMarker" type="System.String" comment="For␣
↪cmdlet Invoke-KFFileTemplate" default="}%"></entry>
      </section>
      <section name="agents">
        <entry key="agentCommandRetryCount" type="System.Int32" default="3"></
↪entry>
        <entry key="agentCommandRetryDelay" type="System.Int32" default="1000"␣
↪comment="in milliseconds"></entry>
        <entry key="agents" type="kenaflow.AgentConfig" default="">
          <!--
          <![CDATA[
    - Name: Local
      Endpoint: kenaflowagent.example.com
      InstanceId: 00000000-0000-0000-0000-000000000000
      Secret: this-key-is-used-in-workflow-scripts
      Passphrase: this-key-is-used-to-communicate-with-the-agent
          ]]>
          -->
        </entry>
      </section>
      <section name="links">
        <entry key="linkClickAnswer" type="System.String" default="">
          <![CDATA[
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>kenaflow</title>
    <style>
    .loginDiv {
        position: absolute;
        left: 50%;
        top: 50%;
        text-align: center;
        background-color: #108f44;
        width:546px;
        height:265px;
        margin-left: -273px; /*half width*/
        margin-top: -132px; /*half height*/
    }
    /* latin-ext */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/RlBXAIuiO5GvH9-0-JbBlw.woff2) format('woff2');
      unicode-range: U+0100-024F, U+1E00-1EFF, U+20A0-20AB, U+20AD-20CF, U+2C60-
↪2C7F, U+A720-A7FF;
    }
    /* latin */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/4hYyXH_8WmbBLamf6WjLwg.woff2) format('woff2');
      unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02C6, U+02DA, U+02DC,␣
↪U+2000-206F, U+2074, U+20AC, U+2212, U+2215;
```

(continues on next page)

(continued from previous page)

```
    }
    p { font-size:40px;font-family:'Dosis';color:white}
    </style>
    </head>
    <div class="loginDiv">
    <p>We got your answer!<br/>Thank you.</p>
    <p><img src="
↪iVBORw0KGgoAAAANSUhEUgAAAMgAAABACAIAAADQ2a98AAABgWlDQ1BzUkdCIElFQzYxOTY2LTIuMQAAKJF1kc8rRFEUxz8
↪4CtspaKSIlZWdNbNBz3oyaSebczj2f+733nO49F6yhlJLWa9yQzuS0gN/
↪rnF9YdNY900AnrVixhxVdnQlOhKhqH3dYzHjTb9aqfu5fa4zGdAUs9cJjiqrlhCeFp9dyqsnbwu1KMhwVPhXu0+SCwremHi
↪HhxMsU4PoYZZFTmYfrxMCArquS7i/mzZCVXkVklj8YKCZLk6BN1VarHJMZFj8lIkTf7/
↪7evenzIU6ru8ELtk2G89UDdFnwXDOPz0DC+j8D2CBeZcn72AEbeRS+UNdc+NG/
↪A2WVZi+zA+SZ0PKhhLVyUbOLWeBxeT6BpAdquwb5U6tnvPsf3EFqXr7qC3T3olfPNyz8ajGfD2mVUoQAAAAlwSFlzAAAuIw
↪dgAADx1JREFUeJztnWlYU8cax+9+EQIKEfgxsi+yKagoyEIG2ig7dXaVmvrrrUtre2+1VXut12Jb69XW0udW66PXWhUrBUU
↪zNn5p0JUNS/
↪8wMMRt5Qp9oBzF8TLCwMKWBhYUgBCwtDClhYGFLAwsKQAhYWhhSwsDCkQBtX6f12wUEOgUTy1P1foxoy5O0S5q/
↪A+ISlqaxhbmBFJNXoKvL2B/
↪MXAX8KMaSAhYUhBSwsDClgYWFIAQsLQwpYWBhSwMLCkAIWFoYUsLAwpPAqCosKEKbJClQ3nmpHMBPn1RKWhSJjl6Xv/
↪VXf//Lmz3baZlPtzhSjTVWYRnm1HtDYGd9aIUkoAmW5jkW47VI/
↪xwAlGn3C7SxS0T4YuIdIbk74V66oSx4OTipUgC0z3F53jbA2su3t7ytqyC9oKkxtyKRRFHYu/
↪pAoFhL7T55kYAr9HJ0pFpY9XWWVmVf47LCZunLonxgKirbGs4ik0p/zdT/
↪qsvYNj7eQTVNQnG+5cL7lwu7rTyrb62SvjkqhTJGDY2JqhMWgUFbr24XZBSy296P+OR8/
↪SQRrTCdUJUtuawlTcdrk+zNhJltY85TVV1j4hDmHGWlNn+RT/
↪yn4m30YYUukkuTCG21drXZGs1IEjaG61lPo2HiZJGFpUBRWGTmG2wd52HhNzhn/
↪pJjqDQ0Jfn9w4d0Hp6bQmZeBdGF5MnVWWC4OcQ7TUdMbe62m9kZOZ6sc3bBRUvbWGxqgZPEqsno6kU0FmKOsZq/
↪OstBiqTJUa3j1FYL6rO5W7kC/bAu2StO89OyQ/UjIjefXA4C5ImO+5kxbHXMVpWlFrRV3uGU1/
↪b3DTq1BUfDSNDFhGpioGzDpzObOtubuNja3vELcM9JPlq4pYec3l07gSlk0pdlqRlZapiwNo6bO1sr2upwOTrVYRBRYrWOp
↪LB011zz+SiZUZl2NikH2fZ0lX97vL/YfumwKtzO1q+uHTrZnE/k2DINvl55ANnxWbEp6ce/
↪c9uwct5q2Vr13Np34j4hJGtCU3rTzOsNtzWGmsNjcl2izjP3fokqS3o0IEY5XkxdZarSNJlNuToqmgFqxgDQLu7O7Ol44WU
↪ENKvPmL+8CgCOdefS1w2hA3PFEEH9y7ejOyF9Y/mqG4Va+wbND1adpjLFKPbc2ueRWct39JAFHIm9/
↪qACH5r9FqKq2rXrrzW+6pBIAWKNnExn2haaK1shaumr6h1d/6/kw5t30n0dKnMlgRvt/
↪hp6ELCxd06+XfOKb8DlKHl601c8x4LleqTLUNvtuZWnNWMeOQpe8b8nHzjNdZMtsXzr4+OMexmSmHx/
↪9Mq0UlY8F7pltOnfkISqFutH7PQ9Ljw8S92b2dJyvvrOlfzMSX5BzqHZONE8yEFeQiC7Hy85HN/
↪2nJSaD7ZQ2FSfw6wFguakHMc2Kz7lEvA9/hJyF5c3UvbD+9BgL13FrbhXfSq67f72jSe56IvjEwidi/
↪uvI5gvbP4jfUyUWAYAZjb4veDehqv6B/
↪rult1s6WgOcArWYOihz2dyI4payb2vuDmtzZA9H4Go+301ZI6NHAAAXC68gYeXVZec15Na0Nxqp6btbLHSY4YwKB88J21Sb
↪nZF9oLtjR0aKnbsBkqAbr2vrYeKNal/
↪LiJWgtxCmUaOq3susv9EfOwqKMIbhS21Z9s+hmcv39m53N5OkJ8ZqO1c6gncjuH+jfcWkPW8hDyc9c1uqq6SOb180N/
↪/W9ol4hAGhkn/vR4/
↪3gOYOzs22+266fzino7R7ZeG5t1qnMsxSgbHB7w4k1h8i31WAhYcU8rlz6IPp2TXp0WxlxpdTCSxmrj1oZ2qDkIrMFLy+s1
↪kCVosGXqfuSxyfVpb+1u7bW5JDmqISOmNIkoHzrL/
↪zgnu0squZx7+V3vTQCwxiUCSb+3X3SxLh0AQjRZ5gaWqHxaeSpxD0dh8sIN1S2Vt0pu3qrPSO5sIVtPCDOmQWTYF0QHHpkQ
↪P3imQkrHlPh8VJuQloLhJqtYRo/2L+ldGdQUySsKRSaRGnoPRxTZ7w8eSoCgD2Be/
↪WYmojO+5hzBEZccxT0abTGETSabrjOe2ZsnW7RV1MhiqyLXRMYYSwDqcdR6oCgBR+Q09fj7KSMkrSqEN31VVZbaPDcj+Hpdj
↪6WcuG1+g2tB1sZ2VAAJQExuHBIWhUJZbuaZVZyQ0SO4V5YqGw+KL7sFAPoKtADnIJTTyGuIbikci0uTJCwKhRLuujLcdeWX
↪9qanV6fXez4TTdINslPrPk9hc6JQDN/EdEAEyDrjqsgJayOmGLB/oa+noAoHWg/
↪3J23HqvjQDA0jVdpmNeK2wjRmMIP4tFZ9tKQ+0DiLrR1XfG6JWchXWvi/
↪v+uW3hdv6+Dv40BcVRStJpDF8Hf18H/30D4vSKeykVd5Me5ZT1PSdmOGG6Rd3/
↪TozcHbgT9RDBc8I31WYc42QDQG4HR7bkjphPTzzKk+OpAcBT30FDRRPZ+fU5a+98Pxi24DfSKFRCWCgsOQHoMhUrWyoIYbl
↪Z0ZHzArAYpOe55a+nHvU4HnEo6VB1S+ULq9AUFD1tffaG703/
↪e9y1oL07LXycGUy5OLPr8ufHOdlHbn5P5Hwa8AlqvEosKmsqIfJXOIYwZOazs+gqMYv/
↪ETkrNEyTxZjoJgIzrRmEzeE9E+/1sfAgbCUa3Ud1HGsSBF7aFgCA3MtpHArkzjJx/
↪Np5FcpXBMo+u+AFlkOny6odmoFm9AhSS5KR7e8cFOYUguxzJVdvFiQBgCpD7Uu/
↪HUT5mJKksbtH1qewrK/
↪ny4obBypuLNMyC7f1C3AOlh0sPxcqhepm6e5m6f4ZQG7twyPs42jZZMKUd3AA4Ghdml95Kor+aapoRXp/
↪uOzaATFID94+enLdT6jkQmvP6wxmUtGNfG65vbbFKteVVoY2vg7+WwDYZXfCrx+YwISjjj/
↪UKQbODt3Pq0vj5OlN01huHzTsO3g0/IDDrxtf2KCw/5nu/PvV/
↪9lQn6Ojqjv73KYfqm6HNYXYGA+uOL3tuXGpfUBVayVLZ6bsGlFbR8t3BbGyjcQUJnjP8gUAOo2B/
↪ipHdk0mW8jTL7+N4hGzTBxRyaKG/ITxPA5yx1gSgNj2mti0Y2YP/
↪hcxc+Fyp1Db6fZjqTjbdK5l0TV4OWERPuxLjYpluTAZTABwt/b6uCr9m6qU2PYab/
↪apNxetR8WcWHNkY1EIvrD9K/
↪ZPE5vG3m7O3y7ko68hlULd5vfhtqeHukVdrYIWIjI0xi6xuKtFKpUSkUKaAs3FbB4AzFFWy+rpPJB85MSaKGJIZ6Q1fdj+Ea
↪/eO3pDRczorsnd2NnZk/H0eQfiOR2v+2eTB0A2J5zfk/cv/jC9ufWKmrIf/3spvQnzz/
↪6Qgp6u79IjOztFw3Lr+fWvnVu84m0cW9bqBKLzqX/
↪MjLfTt0EAOL59atOb8itfX6stbql4q0z740cR4qk0pisoT5M2CuMEBwEADEIE/
↪ITiHy+sD2akzkubyd1VigBSBRwEh+cnJ51ZpXJvHCnkGGrYy9Jh7jnWt7Q7eiUCWccqbrDSjPWfrqA4zN97t2yJAlAVENGyp
↪Xp0LCpe3B6/7/
↪mgtKT6zY6rrQ2tNFgahY1FBQ8Ko6pT68Si1K6WvWuH1lgNs9A3aBEZrR3PT9JgTr4trc+GR7g/
↪kf2ubp2zhIbb2Mtk84ngiZ+0/
```

(continues on next page)

(continued from previous page)

```
    </div>
    <body>
    </body>
    </html>
          ]]>
        </entry>
        <entry key="linkClickExpired" type="System.String" default="">
          <![CDATA[
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>kenaflow</title>
    <style>
    .loginDiv {
        position: absolute;
        left: 50%;
        top: 50%;
        text-align: center;
        background-color: #108f44;
        width:546px;
        height:265px;
        margin-left: -273px; /*half width*/
        margin-top: -132px; /*half height*/
    }
    /* latin-ext */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/RlBXAIuiO5GvH9-0-JbBlw.woff2) format('woff2');
      unicode-range: U+0100-024F, U+1E00-1EFF, U+20A0-20AB, U+20AD-20CF, U+2C60-
↪2C7F, U+A720-A7FF;
    }
    /* latin */
    @font-face {
      font-family: 'Dosis';
      font-style: normal;
      font-weight: 400;
      src: local('Dosis Regular'), local('Dosis-Regular'), url(https://fonts.
↪gstatic.com/s/dosis/v6/4hYyXH_8WmbBLamf6WjLwg.woff2) format('woff2');
      unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02C6, U+02DA, U+02DC,␣
↪U+2000-206F, U+2074, U+20AC, U+2212, U+2215;
    }
    p { font-size:40px;font-family:'Dosis';color:white}
    </style>
    </head>
    <div class="loginDiv">
    <p>The link has expired!<br/>Sorry.</p>
    <p><img src="
↪iVBORw0KGgoAAAANSUhEUgAAAMgAAABACAIAAADQ2a98AAABgWlDQ1BzUkdCIElFQzYxOTY2LTIuMQAAKJF1kc8rRFEUxz8
↪4CtspaKSIlZWdNbNBz3oyaSebczj2f+733nO49F6yhlJLWa9yQzuS0gN/
↪rnF9YdNY900AnrVixhxVdnQlOhKhqH3dYzHjTb9aqfu5fa4zGdAUs9cJjiqrlhCeFp9dyqsnbwu1KMhwVPhXu0+SCwremHi
↪HhxMsU4PoYZZFTmYfrxMCArquS7i/mzZCVXkVklj8YKCZLk6BN1VarHJMZFj8lIkTf7/
↪7evenzIU6ru8ELtk2G89UDdFnwXDOPz0DC+j8D2CBeZcn72AEbeRS+UNdc+NG/
↪A2WVZi+zA+SZ0PKhhLVyUbOLWeBxeT6BpAdquwb5U6tnvPsf3EFqXr7qC3T3olfPNyz8ajGfD2mVUoQAAAAlwSFlzAAAuIwA
↪dgAADx1JREFUeJztnWlYU8cax+9ig9oGrVarUtre2+OR9/Pf+1VXut12dNn1VXut12dJb69XW0udW5uZ2ZFj7j2CZBVmvrrUtre2+OSO
↪zNn5p0JUNS/
↪8wMMRt5Qp9oBzF8TLCwMKWBhYUgBCwtDClhYGFLAwsKQAhYWhhSwsDCkQBtX6f12wUEOgUTy1P1foxoy5O0S5q/
↪A+ISlqaxhbmBFJNXoKvL2B/
↪MXAX8KMaSAhYUhBSwsDClgYWFIAQsLQwpYWBhSwMLCkAIWFoYUsLAwpPAqCosKEKbJClQ3nmpHMBPn1RKWhSJjl6Xv/
↪VXf//Lmz3baZlPtzhSjTVWYRnm1HtDYGd9aIUkoAmW5jkW47VI/
↪xwAlGn3C7SxS0T4YuIdIbk74V66oSx4OTipUgC0z3F53jbA2su3t7ytqyC9oKkxtyKRRFHYu/
↪pAoFhL7T55kYAr9HJ0pFpY9XWWVmVf47LCZunLonxgKirbGs4ik0p/zdT/
↪SQYYiJ7eQFUWQsGuFcL7lwu7rTyrb62SvjkqhTJGDY2JqhMWgUFbr24XZBSy296P+OR8/
↪SQRrTCdUJUtuawlTcdrk+zNhJltY85TVV1j4hDmHGWlNn+RT/
↪yn4m30YYUukkuTCG21drXZGs1IEjaG61lPo2HiZJGFpUBRWGTmG2wd52HhNzhn/
↪pJjqDQ0Jfn9w4d0Hp6bQmZeBdGF5MnVWWC4OcQ7TUdMbe62m9kZOZ6sc3bBRUvbWGxqgZPEqsno6kU0FmKOsZq/
↪OstBiqTJUa3j1FYL6rO5W7kC/bAu2StO89OyQ/UjIjefXA4C5ImO+5kxbHXMVpWlFrRV3uGU1/
```

(continues on next page)

(continued from previous page)

```
    </div>
    <body>
    </body>
    </html>
          ]]>
        </entry>
      </section>
      <!--
      <entry ... overwrite="..">
       can be used to force an value update that replaces the changes probably␣
↪made by users!
    -->
      <section name="donttouchthissection">
        <entry key="instanceId" type="System.Guid" comment="do not change!!"␣
↪default="{00000000-0000-0000-0000-000000000000}"></entry>
        <entry key="configFileVersion" type="System.String" comment="do not change␣
↪this!" default="4.0"/>
        <entry key="productVersion" type="System.String" comment="do not change␣
↪this!" default="4.0.23.4047"/>
        <entry key="val1" type="System.Boolean" default="false" comment="please␣
↪dont touch this"></entry>
        <entry key="val2" type="System.Boolean" default="false" comment="please␣
↪dont touch this"></entry>
        <entry key="crt" type="System.String"></entry>
      </section>
    </config>
```

CHAPTER

SEVENTEEN

WORKFLOW CONFIGURATION

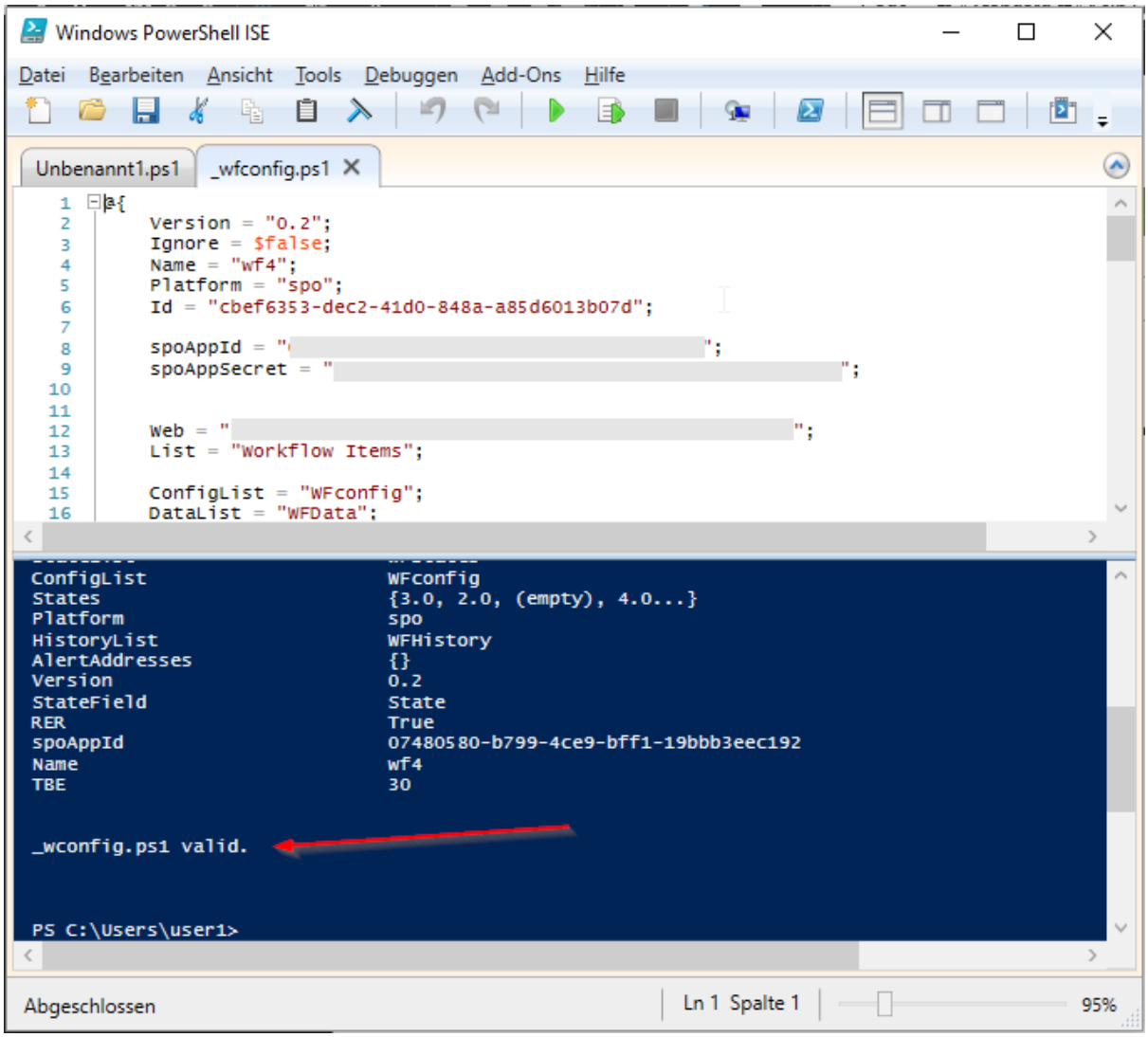
In this section, we introduce you to the possible settings in the workflow configuration file `_wfconfig.ps1`.

Each workflow must have its own configuration file.

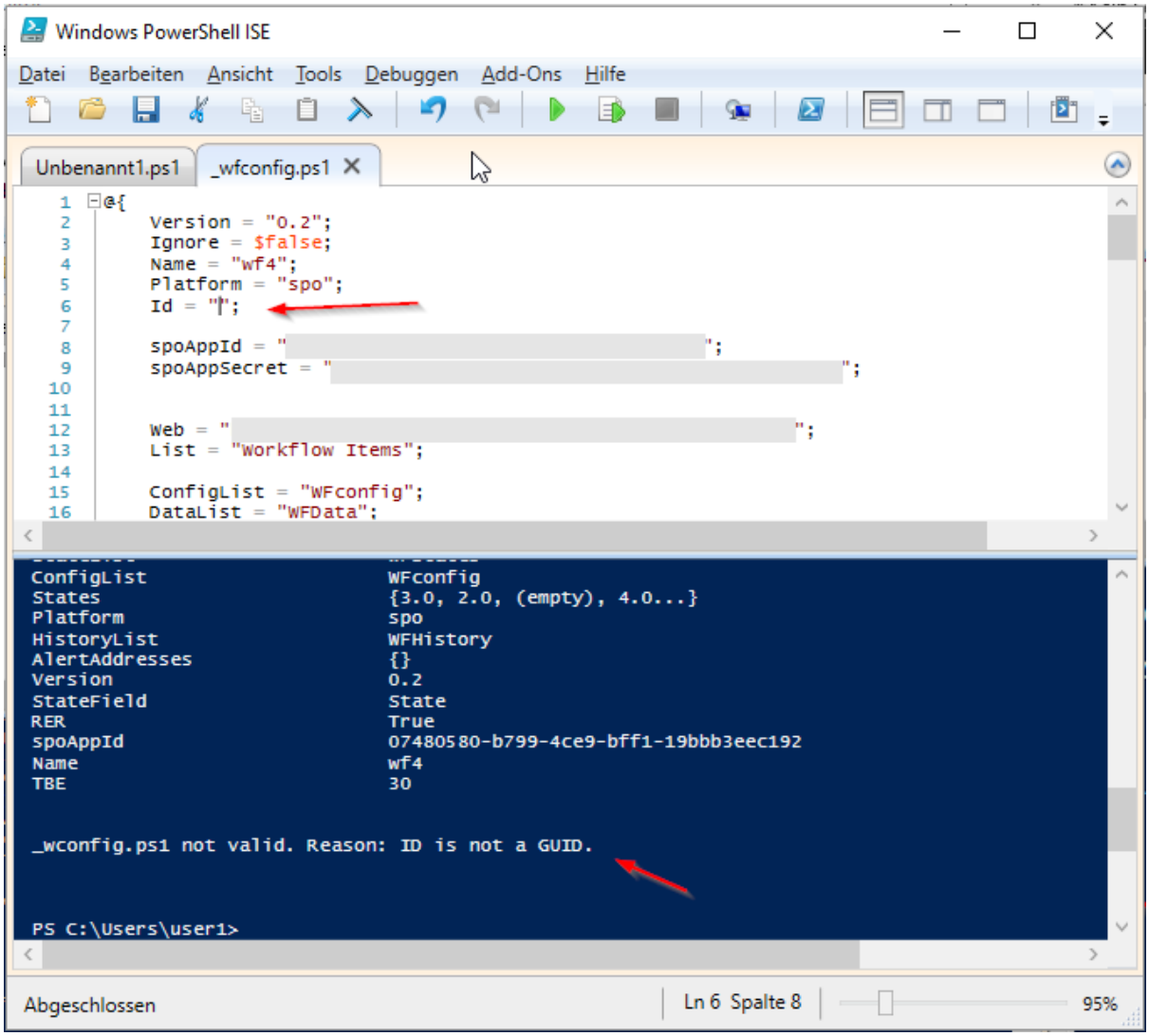
The name `_wfconfig.ps1` is fixed and can not be changed.

The file `_wfconfig.ps1` is always a valid PowerShell script. This basically allows the format of the file to be PowerShelled. That means it can be tested e.g. in the "PowerShell ISE".

The `_wfconfig.ps1` contains a command at the end of the file that starts a *kenaflow* check of that file when the `_wfconfig.ps1` is executed as a script in PowerShell.



Or:



Apart from this check, the execution of the file in PowerShell directly, outside of *kenaflow*, is without effect and harmless - insofar as you have not implemented any code in this file.

It is important to understand that the configuration file `_wfconfig.ps1` is a **script**, not a mere text file. So there can be executable code (PowerShell) in the file. During the execution of the workflow, the configuration file is read and **executed**. For example, dynamic workflow configurations can be created during execution. Please read *Advanced Workflow Configuration* for details.

Some settings in the workflow configuration file are identical for all workflow types.

They are documented here: *General Workflow Settings*

Other settings are special to the workflow type.

- *Site workflow Configuration*
- *List workflow Configuration*
- *State machine workflow Configuration*
- *E-Mail Workflow Configuration*
- *PowerShell Workflow Configuration*

17.1 Most Important Settings to Start With

Following settings are included in every workflow configuration file `_wfconfig.ps1`:

- `Version` [String] This is the version of *kenaflow* for which the workflow was built. This should only be changed if a workflow has been migrated to a new version of *kenaflow*.
- `Enabled` [Boolean] If this value is set to $false, then the workflow will not be executed, unless the option `--noignore` is used with kenaflow.exe.
- `Name` [String] Name of the workflow.
- `Id` [Guid] This value specifies the unique ID of the workflow. This value must be unique at least in the *kenaflow* installation!

17.2 General Workflow Settings

Here are all workflow type independend ("general") workflow settings.

`Version = "4.0";`

A workflow will only work with the following version of kenaflow. After you tested your workflow with a new version of kenaflow you can change the following setting.

`Enabled = $false;`

This is used to enable or disable the workflow.

`$true` means: "Run the workflow" ==> ENABLED `$false` means: "Do not run this workflow" ==> DISABLED

`Slot = 0;`

The execution slot.

Default: `0`

If slots are not configured in global configuration this setting is ignored!

`Priority = 0;`

The execution priority.

Default: `0`

`Type = "";`

This specifies the workflow type. Possible values:

- SHAREPOINT
- EMAIL
- POWERSHELL

`SubType = "";`

This specifies the sub-type of the workflow, depending on the `Type`.

Possible values are:

- `Type` is SHAREPOINT
  - ALL
  - LASTMODIFIED (alias LIST)
  - QUERY
  - SITE

    - STATEMACHINE
  - `Type` is EMAIL
    - EXCHANGE2007SP1
    - EXCHANGE2010
    - EXCHANGE2010SP1
    - EXCHANGE2010SP2
    - EXCHANGE2013
    - EXCHANGE2013SP1
    - EXCHANGE
    - EXCHANGEONLINE
    - POP3
    - IMAP
  - For `Type` POWERSHELL this is always empty.

`Name = "";`

This is the name of the workflow. Technically this will not be used.

`Id = "";`

Each workflow needs an unique ID (System.Guid).

If two workflows have the same ID things will go wrong. One of the workflow will not execute.

You can create such an ID using this PowerShell command: `[guid]::newGuid().tostring('d')`.

`AdditionalConfigFiles = @("..\global_config.ps1")`

With this parameter it is possible to control the serialization process of the workflow configuration. Here you can specify one or more files whose "last write timestamp" are checked during deserialization. If the file(s) where modified the workflow configuration is executed and gets reserialized.

`ScriptFolder = "";`

In 99,99% of all cases the workflow script(s) are in the same folder as `_wfconfig.ps1`.

But if you need to configure it otherwise you can specify the script folder here.

`TBE = 60;`

This means "Time Between Execution".

Between two **starts** of the workflow the configured amount of seconds must elapse.

You can disable a time based run of a workflow if you set `TBE` < `0`.

! You need to set `TBE` to > 0 to use `Cron` (see below). Because a value < 0 disables time-based runs and that includes `Cron` based runs.

`Cron = "* * * * * *";`

You can specify an CRON-style pattern for the time based execution of the workflow.

This CRON pattern will be calculated **relative** to the last workflow run timestamp.

The setting "30 * * * * *" means: Execute 20 minutes past the full hour - every hour.

The setting "15 8-12 * * * *" means: Execute this 15 minutes past the full hour from 8 o'clock to 12 o'clock.

If the workflow is never run this setting is ignored. The workflow will execute once immediately.

The calculated time for the next execution is not guaranteed by *kenaflow*, because exact scheduling is not possible.

`Permanent = $false;`

Sets the workflow to run in a seperate dedicated process instead of the normal queue based execution mode.

`Locking = $true;`

During a time based workflow run the same workflow should start also on another kenaflow server (in case of scaling). If your really sure that you want to run multiple instances of the same workflow than you can set this to `$false`.

Default is `$true` which means: Only one instance of the workflow IF IT RUNS TIME-BASED.

If a certain workflow is started in a remote event handling scenario it can run multiple times in parallel.

`CustomLibrary = ""; CustomLibrary = @("");`

You can specify one or more custom PowerShell library here. They are executed before the workflow script is executed.

In the libraries you can initialize variables and provide functions.

You can share such libraries with multiple workflows!

On global level there can be one custom library, configured in global configuration *Section "Basic"*.

If this global custom library is specified it will be executed *before* the workflows custom libraries, if one or more are specified.

`ErrorHandlingScript = "errorHandling.ps1"`

Here you can specify a script for error handling.

It's processed in case of exceptions within the workflow script.

Please read this artice for details *Error Handling*.

`PreProcessingScript = "preProcessing.ps1"`

This script is executed after the workflow is started. Before any connection, e.g. to SharePoint, is initiated.

The script start with this line to receive the correct parameters:

```
> param($wf, $eventData)
```

`PostProcessingScript = "postProcessing.ps1"`

This script is executed before the workflow is finished.

The script start with this line to receive the correct parameters:

```
> param($wf, $eventData)
```

`Debug = $false;`

You can mark a workflow as being "in debug".

You can evaluate the debug state with Cmdlet *Get-KFInDebug*.

This can be used to have script portions that are only executed during debug.

Default: `$false`.

`NoSerializing = $false;`

The file `_wfconfig.ps1` is a PowerShell script. It is executed when it was changed.

By default the resulting configuration object is serialized for performance reasons.

If you do not which *kenaflow* to serialize the configuration object and *always* evaluate `_wfconfig.ps1` then you set this setting to `$true`.

Default: `$false`

`MaxLifetimeOfSerialization = <value>;`

The workflow configuration is a script and its return value is the workflow configuration object containing all the settings. For performance reasons this configuration object is serialized. The workflow config is not executed as long as it is not changed or *kenaflow* is not updated. You can stop the serialization with property `DoNotSerialize`. But this is a significant loose of performance. Therefore you can use the property `MaxLifetimeOfSerialization` to specify the maximum lifetime of the serialization. `<value>` can be a `System.TimeSpan` object or a timespan as string or an amount of seconds as `System.Int` or an amount of seconds as number in a simple string. This is equivalent:

- `MaxLifetimeOfSerialization = [System.Timespan]"0:1"`
- `MaxLifetimeOfSerialization = "0:1"`
- `MaxLifetimeOfSerialization = "60"`
- `MaxLifetimeOfSerialization = 60` All of these examples stand for `1 minute` or - to be precise - for `60 seconds`.

At time of serialization the next point in time for re-serialization is set and stored in the serialized configuration object.

`RunCondition = @{ConfigKey='<a-key>'; ConfigValue='<a-value>'}`

This can be used to run or stop the workflow in an early state before items are processed.

You can specify a key as `<a-key>` from the workflow config list (in case of SharePoint workflows) or a key from the collection `ConfigListDefaults` (for all workflow types). The value of the key must match the `<a-value>` setting of `RunCondition`. This is case sensitive!

If the value does not match the workflow will be stopped.

In case of multi-web SharePoint workflows the comparison is made for each web. If the webs config value for the given key does not match the `<a-value>` setting the web processing will be skipped. The next web is getting processed.

`StopWorkflowOnScriptError = $true;`

Here you configure if workflow scripts should be stopped if errors (exceptions) occur during execution.

Default is `$true`.

`ConfigListDefaults = @{ "kenaflow"="cool" };`

kenaflow workflows can use key-value-pairs for configuration. For SharePoint workflows and Email workflows with SharePoint connection these settings are stored in a SharePoint list (see property `ConfigList` above) where process owner (typically not workflows developers) can edit the config values, such es email templates.

Here you specify the default values. They are written to SharePoint when creating the config list with command line switch `--createconfiglist`.

In case of Email workflows (without SharePoint connection) and PowerShell workflows there is no SharePoint based config list. In this case the content from this setting is passed to the workflow script as config object.

`AlertAddresses = @();`

In case of workflow errors the configured users will get an email.

Here you specify one or more email.

```
> AltertAddresses = @("developer1@example.com", "delevoper2@example.com")
```

`AlertFloodProtection = 60;`

If `AltertAddresses` are configured (setting above) than the configures users will receive mails in case of workflow errors.

Here you can configure an amout of seconds between two of such error mails. This can help to prevent email flood.

Default: no flood protection.

`MailFrom = "";`

Address of the sender of mails that the workflow sends.

If empty (default) it will be taken from the kenaflow global config.

`MailReply = "";`

Address of the "reply-to" address of mails that the workflow sends.

If empty (default) it will be taken from the kenaflow global config.

`MaxExecutionTime = -1;`

After this amount of time elasped the workflow execution will be stopped

Default: `-1`

`-1` = use default from global configuration.

`ThrowErrorsPreference = $true;`

If the execution of a *kenaflow* sepcific Cmdlet fails an exception will be thrown by default. This causes the workflow script to stop.

The default is `$true`.

If you set this setting to `$false` you can supress exceptions.

If an exception occures it can be read by using cmdlet *`Get-KFLastResult`*.

`ErrorActionPreference = 'Stop';`

PowerShell commands such as cmdlets can fail during execution. PowerShell has a default error handling mechanism that allows the PowerShell runtime to coninue its work or handle errors in a different way, e.g. asking the user to continue or stop.

With this setting you can specify the default action of PowerShell in case of errors in the workflow script.

- `Stop` : Stop the script

- `Continue` : Report the error and continue the script execution if possible
- 'SilentlyContinue` : Do not report the error and continue silently if possible

This default handling can be overwritten when execution the script in the debugger by setting `-ErrorActionPreference` parameter of `Invoke-Kenaflow`.

`writeMailsToDiskDuringDebug = $true;`

When running a workflow script in a debugger such as Visual Studio Code or Windows PowerShell ISE all mails are written to disk by default. The folder is configured here: Global Configuration setting `mailsOnDisk` of *Section "MailDev"*

A workflow developer should first configure `redirectAllMails` (description below) and then set `writeMailstoDiskDuringDebug` to `$false`.

Default is `$true`.

`alwaysWriteMailsToDiskInsteadSending = $false;`

You can force *kenaflow* to write all mails to files in the file system. This can be used for debug or in case of troubleshooting.

Mails are written to the folder configured in Global Configuration setting `mailsOnDisk` of *Section "MailDev"*

`redirectAllMails = @();`

During workflow development all emails should be redirected to the workflow developer. With this setting you can specify one or more email addresses.

ALL mails the workflow sends will be redirected to this address(es).

`SendOutputToMainProcess = $false;`

When writing objects to the PowerShell pipeline they will in the last level always be caught by *kenaflow* runtime environment if no other PowerShell command captures them.

When *kenaflow* receives such objects they are written to the log IF this setting is `$true`. Default is `$false`, so nobody will see the objects.

`MaxExecutionTime = -1;`

After this amount of time elasped the workflow execution will be stopped;

-1 = use default from global configuration. Default there is 600 seconds.

`ScriptFolder = "";`

Can be used to configure the folder where the workflow scripts are.

By default its the same folder that `_wfconfig.ps1` is in.

`FileTemplateStartMarker = "";` and `FileTemplateEndMarker = "";`

This is used for *`Invoke-KFFileTemplate`* and specified the start marker and end marker for placeholders in Word and PowerPoint files.

The default is taken from the global configuration. (Default there is `%{` and `}%`.)

`AllowPowerShellPlaceholders`

This setting is related to *`Invoke-KFFileTemplate`*.

If set to `$true` Word and PowerPoint templates can contain placeholders of the style `<<...>>` that are executed as PowerShell code during placeholder replacement.

Default: `$false`

17.3 Site Workflow Configuration

The workflow settings described in this section belong to *SharePoint Site Workflows*.

`Type = "SHAREPOINT";`

This specifies the workflow type. This must be `SHARPEOINT` in order to be an email workflow.

`SubType = "SITE";`

This specifies the SharePoint workflow sub type. This must be `SITE` in order to be an site workflow.

`Platform = "";`

You need to specify the used SharePoint platform here:

- `sp2013`
- `sp2016`
- `sp2019`
- `spo`

`Environment = "";`

This can be used to specify the SharePoint Online hosting environment:

- `production` : Default Office365 environment
- `usgovernment` : Office365 for US Government
- `germany` : Offic365 in the "German Cloud"
- `china` : Office365 in China Cloud
- `ppe` : ?

`Platform` must be `spo` to use this setting!

`Connection = <object>`

You can execute a SharePoint workflow on multiple webs (and multiple lists if workflow types *list* and *state machine*). The executed workflow script is always the same. All other settings are identical (names of lists such as error list or data list). If you configure this property you cannot configure `Web` and `AlternateWebUrls` in parallel. The `<object>` looks like this: The `<object>` looks like this:

```
> Connection = @(
>   @{
>     Web = "https://firstweb.sharepoint.farm";
>     AlternateWebUrls = @();
>   },
>   @{
>     Web = "https://secondweb.sharepoint.farm";
>     AlternateWebUrls = @();
>   }
> )
```

```
For a description of `Web` and `AlternateWebUrls` please read the␣
↪descriptions of the properties below.
```

`Web = "";`

Full URL to the SharePoint web the workflow is connected to.

Do not specify this setting if you have configured `Connection`!

`RetryOnVersionConflict = $false;`

Force workflow script to run again in case of "version conflict" (parallel item modifications in SharePoint).

Default: `$false`

`UserProfileCacheLifetime = 60;`

With *`Get-KFUserInfo`* a workflow script can query the SharePoint user profile store.

To reduce the calls in case of querying the same account during the workflow run the requested profiles are cached.

Default: global configuration. There: 60 seconds.

`spUser = "";` and `spPwd = '';`

The following two parameters can be used to specify credentials for the workflow for SharePoint access.

If not specified the executing user will be used to access SharePoint.

This user need to have permissions on all object the workflow need to access, e.g. workflow list, items, data list, config list, ...

However it is possible to store credentials in a protected fashion using the parameter *`--setusername`* on `kenaflow.exe`. This will write the encrypted credentials to these two parameters. - Encryption of credentials uses the *kenaflow* encryption passphrase configured in global configuration. - If encrypted the setting contains a Base64 encoded string that starts with unicode character 0xFFFF. This means that the plain `spUser` and `spPassword` are not allowed to start this character!

`spoAppId = "";` and `spoAppSecret = '';`

Only if platform is 'spo' = SharePoint Online

Here you can specify "app credentials" to access SharePoint online.

Please read article *App ID & App Secret*.

You can store encrypted credentials here with program switch *`--appid`* on `kenaflow.exe`. - If encrypted the setting contains a Base64 encoded string that starts with unicode character 0xFFFF. This means that the plain `spUser` and `spPassword` are not allowed to start this character!

`AlternateWebUrls = @();`

A list of alternate URLS, e.g. if a web application has Alternate Access Mappings.

Remove Events could be received with such an alternate URL.

Do not specify this setting if you have configured `Connection`!

`ConfigList = "WFconfig";`

Defines the name of the workflow *config list*.

`DataList = "WFData";`

Defines the name of the workflow *sata list*

`HistoryList = "WFHistory";`

Defines the name of the workflow *history list*

`ErrorList = "WFErrors";`

Defines the name of the workflow *error list*

`RER = $false;`

Specified whether the workflow offers "Remote Event" endpoints.

A list workflow can receive SharePoint Remote Events, Custom Data Remote Events and Link Remote Events.

Please read *Remote Event*.

`MaximumRerLifetime = 120;`

Specifies how long a Remote Event is valid before it is skipped forever (if not processed).

Default: configuration in global config. There is default: 120s.

`PostponeFailedRer = 5;`

Specifies how long a a remote event is postponed in case of errors during processing.

Default: configuration in global config. There is default: 5s.

`DumpRerEventDataToLog = $false;`

When remote events are processes this can be used to force *kenaflow* to write the remove event data to the log.

Because is could write critical data to log this is disabled by default.

Its mainly there for development.

Permission Sets Settings

`PermissionSetCreate="";`, `PermissionSetRead="";`, `PermissionSetUpdate="";`, `PermissionSetDelete="";`, `PermissionSetManage="";` and `PermissionSetApprove="";` can be used in `_wfconfig.ps1` to configure the names of *kenaflow* created permission sets for the *kenaflow* specific SharePoint permission cmdlets:

- *Add-KFPermission*
- *Get-KFPermission*
- *Set-KFPermission*
- *Update-KFPermission*

Please read article *Permissions* for details.

17.4 List Workflow Configuration

The workflow settings described in this section belong to *SharePoint List Workflows*.

`Type = "SHAREPOINT";`

This specifies the workflow type. This must be `SHARPEOINT` in order to be an email workflow.

`SubType = "";`

Valid values for a list workflow are:

- ALL
- LASTMODIFIED
- QUERY

`Platform = "";`

You need to specify the used SharePoint platform here:

- `sp2013`
- `sp2016`

- `sp2019`
- `spo`

`Environment = "";`

This can be used to specify the SharePoint Online hosting environment:

- `production` : Default Office365 environment
- `usgovernment` : Office365 for US Government
- `germany` : Offic365 in the "German Cloud"
- `china` : Office365 in China Cloud
- `ppe` : ?

`Platform` must be `spo` to use this setting!

`Connection = <object>`

You can execute a SharePoint workflow on multiple lists in multiple webs. The executed workflow script is always the same. All other settings are identical (names of lists such as error list or data list). But the lists can be configured per web. If you configure this property you cannot configure `Web`, `AlternateWebUrls` and `List` in parallel. The `<object>` looks like this:

```
> Connection = @(
>   @{
>     Web = "https://firstweb.sharepoint.farm";
>     List = @("List 1", "List 2");
>     AlternateWebUrls = @();
>   },
>   @{
>     Web = "https://secondweb.sharepoint.farm";
>     List = "List 3";
>     AlternateWebUrls = @();
>   },
>   @{
>     Web = "https://thirdweb.sharepoint.farm";
>     List = @("List 4", "List 5");
>     AlternateWebUrls = @();
>   }
> )
```

```
For a description of `Web`, `List` and `AlternateWebUrls` please read the␣
↪descriptions of the properties below.
```

`Web = "";`

Full URL to the SharePoint web where the workflow list is in.

Do not specify this setting if you have configured `Connection`!

`List = @("");`

Title (!) of one or more the SharePoint list where the workflow operates on. Each list are called "workflow list".

You can specify more than 1 list. All of them have to resist in the same web. But sometimes it's required to process multiple lists with the same script. There is no need for more than 1 workflow to do that.

You can continue to use a single string: `List = "Custom List";`. That is equal to `List = @("Custom List");`

Do not specify this setting if you have configured `Connection`!

`RetryOnVersionConflict = $false;`

Force workflow script to run again in case of "version conflict" (parallel item modifications in SharePoint).

Default: `$false`

`UserProfileCacheLifetime = 60;`

With *`Get-KFUserInfo`* a workflow script can query the SharePoint user profile store.

To reduce the calls in case of querying the same account during the workflow run the requested profiles are cached.

Default: global configuration. There: 60 seconds.

`spUser = "";` and `spPwd = '';`

The following two parameters can be used to specify credentials for the workflow for SharePoint access.

If not specified the executing user will be used to access SharePoint.

This user need to have permissions on all object the workflow need to access, e.g. workflow list, items, data list, config list, ...

However it is possible to store credentials in a protected fashion using the parameter *`--setusername`* on `kenaflow.exe`. This will write the encrypted credentials to these two parameters. - Encryption of credentials uses the *kenaflow* encryption passphrase configured in global configuration. - If encrypted the setting contains a Base64 encoded string that starts with unicode character 0xFFFF. This means that the plain `spUser` and `spPassword` are not allowed to start this character!

`spoAppId = "";` and `spoAppSecret = '';`

Only if platform is 'spo' = SharePoint Online

Here you can specify "app credentials" to access SharePoint online.

Please read article *App ID & App Secret*.

You can store encrypted credentials here with program switch *`--appid`* on `kenaflow.exe`. - If encrypted the setting contains a Base64 encoded string that starts with unicode character 0xFFFF. This means that the plain `spUser` and `spPassword` are not allowed to start this character!

`AlternateWebUrls = @();`

A list of alternate URLS, e.g. if a web application has Alternate Access Mappings.

Remove Events could be received with such an alternate URL.

Do not specify this setting if you have configured `Connection`!

`ConfigList = "WFconfig";`

Defines the name of the workflow *config list*.

`DataList = "WFData";`

Defines the name of the workflow *sata list*

`HistoryList = "WFHistory";`

Defines the name of the workflow *history list*

`ErrorList = "WFErrors";`

Defines the name of the workflow *error list*

`PreQueryProcessing = "preQueryProcessing.ps1";`

If specified this script is executed before querying.

Default: no pre-query script.

Please read *Workflow Scripts*.

```
RER = $false;
```

Specified whether the workflow offers "Remote Event" endpoints.

A list workflow can receive SharePoint Remote Events, Custom Data Remote Events and Link Remote Events.

Please read *Remote Event*.

```
MemBasedRer = $false;
```

In case of a "permanent" running workflow this setting can be used to process remote events in a memory based queue instead of a file based queue.

```
AllowParallelRer = $false;
```

In case of a "permanent" running workflow ... By default (`$false`) remote events are executed one by one.

When setting it to `$true` remote events can be executed in parallel.

```
RREEvents = @();
```

List of strings that contain the subscribed remote events

Valid strings:

- `ITEMADDED`
- `ITEMUPDATED`

Default: Both.

```
RERHandlerName = "kenaflow_wf1";
```

Optional. Used as prefix for the name of the SharePoint Remote Event Receiver.

The default name is configured in the global configuration. It is "kenaFlow".

```
MaximumRerLifetime = 120;
```

Specifies how long a Remote Event is valid before it is skipped forever (if not processed).

Default: configuration in global config. There is default: 120s.

```
PostponeFailedRer = 5;
```

Specifies how long a a remote event is postponed in case of errors during processing.

Default: configuration in global config. There is default: 5s.

```
LastModifiedBasedOnLastRun = true;
```

If `$true`:

- the timestamp `LastRun` of the workflow will be used to identify "last modified" items
- the queried items for processing are sorted by ID.

If `$false`:

- a time stamp file is written after processing all items or
- a time stamp file is written if the workflows is timed out and forced to stop and
- the queried items for processing are sorted by their "Modified" timestamp.

```
ItemBatchCount = -1;
```

Count of list items queried from SharePoint at one request.

Default: `-1`

`-1` = use default from global configuration. Default: 1000.

`ItemReCheckTime = -1;`

Amount of time before each item is checked to be processed.

Default: `-1`

- `-1` = use default from global configuration. Default: 60 seconds;
- `0` = check immediately

Background: If you query 1000 items from SharePoint to be processed by your workflow it could take minutes until item #200 is processed. Maybe the item does not need processing at this point in time because e.g. it was changed since them and would not match the query if it would be requested now. - Thats what *kenaflow* does. It "re-checks" all follwoing items to be "in query" right before executing them.

`IgnoreFoldersInQuery = $false;`

Handle lists and libraries "flat" by ignoring folders.

Default: `$false`.

`Query="...";`

This uses by default the *kenaflow* query language: *Queries*.

However if the query string starts with "<" it will be treated as CAML.

`ItemChangeHashFields = <value>;`

If you only want to process item with changes in certain item fields (columns) you can specify the list of names here.

If you want only process items with changes in any field you can set this simply to $true.

Example:

```
> ItemChangeHashFields = @("Title");
```

`DumpRerEventDataToLog = $false;`

When remote events are processes this can be used to force *kenaflow* to write the remove event data to the log.

Because is could write critical data to log this is disabled by default.

Its mainly there for development.

Permission Sets Settings

`PermissionSetCreate="";`, `PermissionSetRead="";`, `PermissionSetUpdate="";`, `PermissionSetDelete="";`, `PermissionSetManage="";` and `PermissionSetApprove="";` can be used in `_wfconfig.ps1` to configure the names of *kenaflow* created permission sets for the *kenaflow* specific SharePoint permission cmdlets:

- *Add-KFPermission*
- *Get-KFPermission*
- *Set-KFPermission*

- *Update-KFPermission*

Please read article *Permissions* for details.

17.5 State Machine workflow Configuration

The workflow settings described in this section belong to *SharePoint State Machine Workflows*.

`Type = "SHAREPOINT";`

This specifies the workflow type. This must be `SHAREPOINT` in order to be an email workflow.

`SubType = "STATEMACHINE";`

This specifies the SharePoint workflow sub type. This must be `STATEMACHINE` in order to be an state machine workflow.

`Platform = "";`

You need to specify the used SharePoint platform here:

- `sp2013`
- `sp2016`
- `sp2019`
- `spo`

`Environment = "";`

This can be used to specify the SharePoint Online hosting environment:

- `production` : Default Office365 environment
- `usgovernment` : Office365 for US Government
- `germany` : Offic365 in the "German Cloud"
- `china` : Office365 in China Cloud
- `ppe` : ?

`Platform` must be `spo` to use this setting!

`Connection = <object>`

You can execute a SharePoint workflow on multiple lists in multiple webs. The executed workflow script is always the same. All other settings are identical (names of lists such as error list or data list). But the lists can be configured per web. If you configure this property you cannot configure `Web`, `AlternateWebUrls` and `List` in parallel. The `<object>` looks like this:

```
> Connection = @(
>   @{
>     Web = "https://firstweb.sharepoint.farm";
>     List = @("List 1", "List 2");
>     AlternateWebUrls = @();
>   },
>   @{
>     Web = "https://secondweb.sharepoint.farm";
>     List = "List 3";
>     AlternateWebUrls = @();
>   },
>   @{
>     Web = "https://thirdweb.sharepoint.farm";
```

(continues on next page)

(continued from previous page)

```
>     List = @("List 4", "List 5");
>     AlternateWebUrls = @();
>   }
> )
```

```
For a description of `Web`, `List` and `AlternateWebUrls` please read the␣
↪descriptions of the properties below.
```

`Web = "";`

Full URL to the SharePoint web where the workflow list is in.

`List = @("");`

Title (!) of one or more the SharePoint list where the workflow operates on. Each list are called "workflow list".

You can specify more than 1 list. All of them have to resist in the same web. But sometimes it's required to process multiple lists with the same script. There is no need for more than 1 workflow to do that.

You can continue to use a single string: `List = "Custom List";`. That is equal to `List = @("Custom List");`

The states of each list - if there are more than 1 - comming all from the same state list.

Do not specify this setting if you have configured `Connection`!

`StateList = "";`

The name of the lookup list where the states are stored in. This list can be created with program switch *--createstatelist*. It's a simple "custom list" with a single-line-of-text-field "State" that contains the state key. The default "Title" is used for the human readable state title.

`StateField = "";`

The internal name of the state field that is added to the workflow list with *--addstatefield*. Its a lookup field that points to the `StateList`.

Ensure that the name does not contain any special characters! If necessare change the technical name later in SharePoint manually. Do not use spaces etc.

`RetryOnVersionConflict = $false;`

Force workflow script to run again in case of "version conflict" (parallel item modifications in SharePoint).

Default: `$false`

`UserProfileCacheLifetime = 60;`

With *Get-KFUserInfo* a workflow script can query the SharePoint user profile store.

To reduce the calls in case of querying the same account during the workflow run the requested profiles are cached.

Default: global configuration. There: 60 seconds.

`spUser = "";` and `spPwd = '';`

The following two parameters can be used to specify credentials for the workflow for SharePoint access.

If not specified the executing user will be used to access SharePoint.

This user need to have permissions on all object the workflow need to access, e.g. workflow list, items, data list, config list, ...

However it is possible to store credentials in a protected fashion using the parameter *--setusername* on `kenaflow.exe`. This will write the encrypted credentials to these two parameters. - Encryption of credentials uses the *kenaflow* encryption passphrase configured in global configuration. - If encrypted the setting contains a Base64 encoded string that starts with unicode character 0xFFFF. This means that the plain `spUser` and `spPassword` are not allowed to start this character!

`spoAppId = "";` and `spoAppSecret = '';`

Only if platform is 'spo' = SharePoint Online

Here you can specify "app credentials" to access SharePoint online.

Please read article *App ID & App Secret*.

You can store encrypted credentials here with program switch *--appid* on `kenaflow.exe`. - If encrypted the setting contains a Base64 encoded string that starts with unicode character 0xFFFF. This means that the plain `spUser` and `spPassword` are not allowed to start this character!

`AlternateWebUrls = @();`

A list of alternate URLS, e.g. if a web application has Alternate Access Mappings.

Remove Events could be received with such an alternate URL.

Do not specify this setting if you have configured `Connection`!

`ConfigList = "WFconfig";`

Defines the name of the workflow *config list*.

`DataList = "WFData";`

Defines the name of the workflow *sata list*

`HistoryList = "WFHistory";`

Defines the name of the workflow *history list*

`ErrorList = "WFErrors";`

Defines the name of the workflow *error list*

`PreQueryProcessing = "preQueryProcessing.ps1";`

If specified this script is executed before querying.

Default: no pre-query script.

Please read *Workflow Scripts*.

`RER = $false;`

Specified whether the workflow offers "Remote Event" endpoints.

A list workflow can receive SharePoint Remote Events, Custom Data Remote Events and Link Remote Events.

Please read *Remote Event*.

`MemBasedRer = $false;`

In case of a "permanent" running workflow this setting can be used to process remote events in a memory based queue instead of a file based queue.

`AllowParallelRer = $false;`

In case of a "permanent" running workflow ... By default (`$false`) remote events are executed one by one.

When setting it to `$true` remote events can be executed in parallel.

`REREvents = @();`

List of strings that contain the subscribed remote events

Valid strings:

- `ITEMADDED`
- `ITEMUPDATED`

Default: Both.

`RERHandlerName = "kenaflow_wf1";`

Optional. Used as prefix for the name of the SharePoint Remote Event Receiver.

The default name is configured in the global configuration. It is "kenaFlow".

`MaximumRerLifetime = 120;`

Specifies how long a Remote Event is valid before it is skipped forever (if not processed).

Default: configuration in global config. There is default: 120s.

`PostponeFailedRer = 5;`

Specifies how long a a remote event is postponed in case of errors during processing.

Default: configuration in global config. There is default: 5s.

`ItemBatchCount = -1;`

Count of list items queried from SharePoint at one request.

Default: `-1`

`-1` = use default from global configuration. Default: 1000.

`ItemReCheckTime = -1;`

Amount of time before each item is checked to be processed.

Default: `-1`

- `-1` = use default from global configuration. Default: 60 seconds;
- `0` = check immediately

Background: If you query 1000 items from SharePoint to be processed by your workflow it could take minutes until item #200 is processed. Maybe the item does not need processing at this point in time because e.g. it was changed since them and would not match the query if it would be requested now. - Thats what *kenaflow* does. It "re-checks" all follwoing items to be "in query" right before executing them.

`IgnoreFoldersInQuery = $false;`

Handle lists and libraries "flat" by ignoring folders.

Default: `$false`.

`ItemChangeHashFields = <value>;`

If you only want to process item with changes in certain item fields (columns) you can specify the list of names here.

If you want only process items with changes in any field you can set this simply to $true.

This setting can be overwritten for each state.

Example:

```
> ItemChangeHashFields = @("Title");
```

```
StateSpecificItemChangeHash = $false;
```

If you set this to `$true` than each workflow state will have its own change hash.

The default for this setting is `$false`.

Permission Sets Settings

`PermissionSetCreate="";` , `PermissionSetRead="";` , `PermissionSetUpdate="";` , `PermissionSetDelete="";` , `PermissionSetManage="";` and `PermissionSetApprove="";` can be used in `_wfconfig.ps1` to configure the names of *kenaflow* created permission sets for the *kenaflow* specific SharePoint permission cmdlets:

- *Add-KFPermission*
- *Get-KFPermission*
- *Set-KFPermission*
- *Update-KFPermission*

Please read article *Permissions* for details.

17.5.1 Workflow State Configuration

The power of *state machine* workflows comes from the ability to execute code depending on the current item state.

Each state has a key as technical name and its own configuration.

The state key must not contain characters that are forbidden in file names, because the state key is used for file names, such as the state script file.

All states together are defined in a Hashtable property with the technical state name as key:

```
States = @{
    "<state-key>" : <state-config-hashtable>
    "(empty)" : @{...}
    "State2" : @{...}
    "State3" : @{...}
}
```

The name `(empty)` for the "empty State" - if the workflows state field in SharePoint is `$null` for the list item - is fixed and cannot be changed.

Each state configuration (`<state-config-hashtable>`) has the following options:

```
Enabled = $true;
```

Each state can be disable for processing by setting this to `$false`.

Default is `$true`.

```
Title = "";
```

This is the human readable name of the state. It will be written to the state list, which is a lookup list that gets connected to the workflow list and stores the current state for each item.

```
Type="lastmodified";
```

Valid values are:

- `query` : All items in the current state that match the query.
- `lastmodified` : All items in the current state that were modified sind last running the script.
- `all` : All items in the current state are processed.

Default is: `all`

This is used to query list items from SharePoint. First the state field is always used for the query.

`Query="";`

If `Type` is `query...`

This uses by default the *kenaflow* query language: *Queries*.

However if the query string starts with "<" it will be treated as CAML.

`ExecutionOrder = 100;`

You can define the execution order for states by defining this key.

All states that have not defined this setting the number 0 is used. Except for state "(empty)" for which -1 is used by default. So `(empty)` will be executed first.

All scripts with the same "ExecutionOrder" (or none which means "0") are executed by the order of their keys.

`ItemReCheckTime = -1;`

Amount of time before each item is checked to be processed.

Default: `-1`

- `-1` = use default from workflow configuration.

`MaxExecutionTime = -1;`

After this amount of time elasped the workflow execution will be stopped

Default: `-1`

`-1` = use default from `_wfconfig.ps1`.

`ItemBatchCount = -1;`

Count of list items queried from SharePoint at one request.

Default: `-1`

`-1` = use default from `_wfconfig.ps1`.

`Debug = $false;`

You can mark a workflow state as being "in debug".

You can evaluate the debug state with Cmdlet *`Get-KFInDebug`*.

This can be used to have script portions that are only executed during debug.

Default: `$false`.

`LastModifiedBasedOnLastRun = true;`

If `$true` :

- the timestamp `LastRun` of the workflow state will be used to identify "last modified" items
- the queried items for processing this state are sorted by ID.

If `$false` :

- a time stamp file is written after processing all items for the state or
- a time stamp file is written if the workflows is timed out and the state script is forced to stop and
- the queried items for processing are sorted by their "Modified" timestamp.

`IgnoreFoldersInQuery = $false;`

Handle lists and libraries "flat" by ignoring folders.

Default: `$false`.

`Script = "";`

The name of the state script file.

By default (if not specified) the state script name is `S_` plus the technical state name.

```
> State = @{
>   # Script name is S_(empty).ps1
>   "(empty)" : @{...}
>
>   # Script name is S_State2.ps1
>   "State2" : @{...}
>
>   # Script name is S_State3.ps1
>   "State3" : @{...}
>
>   # Script name is State4.ps1
>   "State4" : @{ ... Script="State4.ps" ... }
> }
```

`TBE = 30;`

This means "Time Between Execution".

Between two **starts** of the state script the configured amount of seconds must elapse.

`Cron = "60 * * * *";`

You can specify an CRON-style pattern for the time based execution of the workflow state.

This CRON pattern will be calculated **relative** to the last state run timestamp.

If the state is never run this setting is ignored. The state script will execute once immediately.

The calculated time for the next execution is not guaranteed by *kenaflow*, because exact scheduling is not possible.

`ItemChangeHashFields = <value>;`

If you only want to process item with changes in certain item fields (columns) you can specify the list of names here.

If you want only process items with changes in any field you can set this simply to $true.

If this setting is not specified for a certain state but is specified for the whole workflow in the corresponding setting `ItemChangeHashFields` than the workflows setting is used for the state. If you want to **disable** this feature for the current state and the global setting of the then you can set this to $false.

If this setting is specified it will overwrite the workflows setting if this is specified.

Examples:

```
> ItemChangeHashFields = @("Title");
> State = @{
>   # This disabled the feature
>   "(empty)" : @{
>      ItemChangeHashFields = $false
>   }
>
```

(continues on next page)

(continued from previous page)

```
>    # This uses the setting of the workflow
>    "State2" : @{
>    }
>
>    # This will use field "CustomField" as reference for the change hash instead
>    #  of the global defined "Title" field.
>    "State3" : @{
>       ItemChangeHashFields = @("CustomField")
>    }
>
>    # This uses all fields for the hash.
>    "State4" : @{
>       ItemChangeHashFields = $true
>    }
> }
```

```
If workflow-level setting `StateSpecificItemChangeHash` is `$true` than␣
↪each state will have its own change hash saved in [the _kenaflow_␣
↪Field](#fd5c3e81-73e7-4228-9a36-e44ff7b6f5a7). Default for this setting␣
↪is `$false`.
```

17.6 Email Workflow Configuration

The workflow settings described in this section belong to *Email Workflows*.

`Type = "EMAIL";`

This specifies the workflow type. This must be `EMAIL` in order to be an email workflow.

`SubType = "";`

This specifies the sub-type of the workflow, depending on the `Type`.

Possible values for an email workflow are:

- EXCHANGE2007SP1
- EXCHANGE2010
- EXCHANGE2010SP1
- EXCHANGE2010SP2
- EXCHANGE2013
- EXCHANGE2013SP1
- EXCHANGE
- EXCHANGEONLINE
- POP3
- IMAP

`EmailUsername = "";`

The access account for the email inbox.

`EmailPassword = "";`

The password for the email inbox.

`EmailIgnoreSSLErrors = $true;`

If set to `$true` (default) SSL errors are ignored when connecting to the inbox.

Specific Settings for IMAP and POP3

`EmailServerAddress = "imap.example.com";`

- Server addess (required for SUBTYPE == POP3 or == IMAP)

`EmailServerSecurity = "AUTO";`

Security Settings. Valid values are:

- `AUTO` (default)
- `NONE`
- `SSLONCONNECT`
- `STARTTLS`
- `STARTTLSWHENAVAILABLE`

`EmailServerPort = -1;`

This specifies the port of the mail server.

`-1` is default: guess 'port' from `_wfconfig.ps1` properties `Subtype` (IMAP, POP3) and `EmailServerSecurity`.

Specific settings for EXCHANCE

`EmailDomain = "example.com";`

For platform "Exchange" (on-premises or Online). Here you configure the mail domain.

`EmailExchangeAutodiscoverEmailAddress = "";`

E-Mail address of the monitored account - e.g. kenaflow@example.com - to auto-discover the Exchange server.

`EmailExchangeServiceUrl = "";`

URL of the Exchange Service.

SharePoint connected Email Workflow

`Platform = "";`

If the workflow is connected to PowerShell you need to specify the used SharePoint platform here:

- sp2013
- sp2016
- sp2019
- spo

`spUser = "";` and `spPwd = '';`

The following two parameters can be used to specify credentials for the workflow for SharePoint access.

If not specified the executing user will be used to access SharePoint.

This user need to have permissions on all object the workflow need to access, e.g. workflow list, items, data list, config list, ...

However it is possible to store credentials in a protected fashion using the parameter `--setusername` on `kenaflow.exe`. This will write the encrypted credentials to these two parameters. - Encryption of credentials uses the *kenaflow* encryption passphrase configured in global configuration.

`spoAppId = "";` and `spoAppSecret = '';`

Only if platform is 'spo' = SharePoint Online

Here you can specify "app credentials" to access SharePoint online.

Please read article *App ID & App Secret*.

You can store encrypted credentials here with program switch `--appid` on `kenaflow.exe`.

```
Web = "";
```

Full URL to the SharePoint web where the workflow list is in.

```
AlternateWebUrls = @();
```

A list of alternate URLS, e.g. if a web application has Alternate Access Mappings.

Remove Events could be received with such an alternate URL.

```
ConfigList = "WFconfig";
```

Defines the name of the workflow *config list*.

```
DataList = "WFData";
```

Defines the name of the workflow *sata list*

```
HistoryList = "WFHistory";
```

Defines the name of the workflow *history list*

```
ErrorList = "WFErrors";
```

Defines the name of the workflow *error list*

```
RetryOnVersionConflict = $false;
```

Force workflow script to run again in case of "version conflict" (parallel item modifications in SharePoint).

Default: `$false`

```
UserProfileCacheLifetime = 60;
```

With *Get-KFUserInfo* a workflow script can query the SharePoint user profile store.

To reduce the calls in case of querying the same account during the workflow run the requested profiles are cached.

Default: global configuration. There: 60 seconds.

17.7 PowerShell Workflow Configuration

The workflow settings described in this section belong to *PowerShell Workflows*.

```
RER = $false;
```

Specified whether the Workflow needs "Remote Event Receiver" to execute the workflow scripts immediately after a change in SharePoint.

```
MemBasedRer = $false;
```

In case of a "permanent" running workflow this setting can be used to process remote events in a memory based queue instead of a file based queue.

```
AllowParallelRer = $false;
```

In case of a "permanent" running workflow ... By default (`$false`) remote events are executed one by one.

When setting it to `$true` remote events can be executed in parallel.

```
#MaximumRerLifetime = 120;
```

Specifies how long a Remote Event is valid before it is skipped forever (if not processed). Default: configuration in global config.

There is default: 120s.

`PostponeFailedRer = 5;`

Specifies how long a a remote event is postponed in case of errors during processing.

By default the setting from global configuration is used. There it has a default of 5 seconds.

`SaveRerEventsAfterProcessing = $false;`

It is possible to save processed remote event files. Normally they are deleted.

Saving can be enabled in the global configuration but must be also enabled for each workflow itself by setting this property to `$true`.

Default: `$false`

17.8 Complete Template

Each workflow has a file `_wfconfig.ps1` with exactly this name.

This file defines the workflow and controls its execution.

...

17.8.1 Complete file `_wfconfig.ps1`

Here you can find a copy of the complete file:

- *Site Workflow Complete Template*
- *List Workflow Complete Template*
- *State Machine Workflow Complete Template*
- *Email Workflow Complete Template*
- *PowerShell Workflow Complete Template*

17.8.2 Site Workflow Complete Template

```
@{
    #* A workflow will only work with the following version of
    #*   kenaflow. After you tested your workflow with a new
    #*   version of kenaflow you can change the following
    #*   setting.
    Version = "4.0";

    #* set it to $true to ENABLE the workflow. ($false DISABLES it)
    Enabled = $false;

    #* Type = SHAREPOINT, EMAIL, POWERSHELL
    Type = "SHAREPOINT";

    #* SubType
    #*   if "SHAREPOINT" then ALL, LASTMODIFIED (alias LIST),
    #*   QUERY, SITE, STATEMACHINE if "EMAIL" then
    #*   EXCHANGE2007SP1, EXCHANGE2010, EXCHANGE2010SP1,
    #*   EXCHANGE2010SP2, EXCHANGE2013, EXCHANGE2013SP1,
    #*   EXCHANGE, EXCHANGEONLINE, POP3, IMAP if "POWERSHELL" => EMPTY
```

(continues on next page)

(continued from previous page)

```
SubType = "SITE";

#* set it to a meaningful name
Name = "<<wfname>>";

#* what SharePoint do you use? Possible values: "sp2013" or
#*   "sp2016" or "sp2019" or "spo"
Platform = "";

#* a unique (!) id (type 'System.Guid') for this workflow.
#*   must be unique in YOUR environment. Use
#*   "[guid]::newGuid().tostring('d')" to create a GUID in
#*   Powershell.
Id = "<<wfid>>";

#* A list of one or more files that are checked during deserialization.
#*   The workflow configuration is reserialized if such an additional
#*   file was modified since last serialization or is missing.
#AdditionalConfigFiles = @();

#* option. can be used to specify the folder where the
#*   workflow scripts are in. the workflow directory (where
#*   the _wfconfig.ps1 is in) is used if empty.
#ScriptFolder = "";

#* "Time between execution" => minimal time difference IN
#*   SECONDS between two executions of the workflow Make sure
#*   you set TBE > 0 to activate time-based execution. First
#*   TBE is checked and afterwards CRON if set.
TBE = 60;

#* CRON-style pattern for execution scheduling. Please
#*   consult our documentation to get more infos on that.
#Cron = "0 * * * *";

#* During a time based workflow run the same workflow
#*   should start also on another kenaflow server (in case of
#*   scaling). If your really sure that you want to run
#*   multiple instances of the same workflow than you can set
#*   this to $FALSE.
#Locking = $true;

#* This can be used to specify one or more custom
#*   PowerShell libariy files that will be executed before
#*   workflow script execution.
#CustomLibrary = @("");

#* Here you can specify a script for error handling. It's
#*   processed in case of exceptions within the workflow
#*   script. Please read
#*   https://doc.kenaflow.com/basics/errorhandling
#ErrorHandlingScript = "errorHandling.ps1"

#* Can be used to mark this workflow as "in debug" for use
#*   with cmdlet Get-KFInDebug -WorkflowDebugState
#Debug = $false;

#* This file will be serialized for performance reasons.
#*   You can prevent serialization by setting this option.
#NoSerializing = $false;
```

(continues on next page)

(continued from previous page)

```
        #* This file will be serialized for performance reasons as
        #*   long as property 'NoSerializing' is not set to $true.
        #*   But you can specify a "maximum lifetime" here. if this
        #*   amount of time elapsed the workflow configuration gets
        #*   re-serialized. Possible values are TimeSpan (object or
        #*   string) or amount of seconds (int or string)
        #MaxLifetimeOfSerialization = "0:5" #5 minutes

        #UserProfileCacheLifetime = 60; #In seconds. If not specified the global␣
↪setting from kenaflow.conf is used

        #* Force workflow script to run again in case of "version
        #*   conflict" (parallel item modifications in SharePoint).
        #*   Default: $false
        #RetryOnVersionConflict = $false;

        #* If set to $false the workflow will continue with the
        #*   next item on workflow script errors. Default ist $true
        #*   ("stop workflow on script errors")
        #StopWorkflowOnScriptError = $true;


        #* The following two parameters can be used to specify credentials
        #*   however it is possible to store credentials in a
        #*   protected fashion using the parameters --setusername
        #*   and --setpassword on kenaflow.exe while running the
        #*   program inside a workflow folder that contains
        #*   _wfconfig.ps1. this will create a file
        #*   _wfconfig-cred.ps1 that contains encrypted credentials.
        #*   the credentials can only decrypted on the same machine.
        #*   on other machines the file is worthless.
        #spUser = "";
        #spPwd = '';

        #* only if platform is 'spo' = SharePoint Online
        #spoAppId = "";
        #* only if platform is 'spo' = SharePoint Online
        #spoAppSecret = '';

        #* You can execute a SharePoint
        #*   workflow on multiple webs (and multiple lists if
        #*   workflow types *list* and *state machine*). The executed
        #*   workflow script is always the same. All other settings
        #*   are identical (names of lists such as error list or data
        #*   list). If you configure this property you cannot
        #*   configure `Web` and `AlternateWebUrls` in parallel.
        <#
        Connection = @(
          @{
            Web = "https://firstweb.sharepoint.farm";
            AlternateWebUrls = @();
          },
          @{
            Web = "https://secondweb.sharepoint.farm";
            AlternateWebUrls = @();
          }
        )
        #>

        #* Full URL to the SharePoint web where the workflow list
        #*   is in
```

(continues on next page)

(continued from previous page)

```
        Web = "";

        #* A list of alternate URLS, e.g. if a web application has
        #*   Alternate Access Mappings. Remove Events could be
        #*   received with such an alternate URL
        #AlternateWebUrls = @();

        #* a list for global config values
        ConfigList = "WFconfig";

        #* a list that can be used to store data
        DataList = "WFData";

        #* a list that can be used to record log information
        HistoryList = "WFHistory";

        #* Workflow errors will be stored in this SharePoint list
        ErrorList = "WFErrors";

        #* Can be uses to define default config entries that will
        #*   be deployed to the config list if the workflow config
        #*   list is created by kenaflow (kenaflow.exe)
        ConfigListDefaults = @{
            "kenaflow" = "cool"
        };

        #* List of mail addresses that will receive mails in case
        #*   of workflow internal errors.
        AlertAddresses = @();

        #* can be used to overwrite the global settings.
        #AlertFloodProtection = 3600;

        #* Address of the sender of mails that the workflow sends.
        #*   If empty it will be taken from the kenaflow global
        #*   config
        #MailFrom = "";

        #* Address to which uses can reply to. If empty it will be
        #*   taken from the kenaflow global config
        #MailReply = "";

        #* This script contains all to workflow actions
        #Script = "script.ps1";

        #* After this amount of time elasped the workflow execution
        #*   will be stopped; -1 = use default
        #MaxExecutionTime = -1;

        #* Specified whether the Workflow needs Remote Event
        #*   Receiver to execute the workflow scripts immediately
        #*   after a change in SharePoint.
        RER = $false;

        #* Specifies how long a Remote Event is valid before it is
        #*   skipped forever (if not processed). Default:
        #*   configuration in global config. There is default: 120s.
        #MaximumRerLifetime = 120;

        #* Specifies how long a a remote event is postponed in case
        #*   of errors during processing. Default: configuration in
```

(continues on next page)

(continued from previous page)

```
        #*   global config. There is default: 5s.
        #PostponeFailedRer = 5;


        #* Some item permission related Cmdlets in workflows can
        #*   simply set "CRUD" +M (Manage) +A (Approve) permissions
        #*   by specifying some characters from set [C, R, U, D, M,
        #*   A]. They refer to permission set that the Site
        #*   Collection administrator has to create. They must also
        #*   be named here
        #PermissionSetCreate="";
        #PermissionSetRead="";
        #PermissionSetUpdate="";
        #PermissionSetDelete="";
        #PermissionSetManage="";
        #PermissionSetApprove="";


        #* ++++++++++++++++++
        #*   runtime settings

        #* (only) kenaflow Cmdlets will throw Exceptions on failure
        #*   ($true) or hide them ($false). In the last case the
        #*   exception can be queried with Get-KFLastResult. Default
        #*   is $true
        #ThrowErrorsPreference = $true;
        #* This is the default for the common PowerShell variable
        #*   $ErrorActionPreference that controls the default
        #*   PowerShell error handling. Default is "Stop". This can
        #*   be overwritten by Invoke-Kenaflow -ErrorActionPreference
        #*   <value> or in script using $ErrorActionPreference =
        #*   <value>
        #ErrorActionPreference = 'Stop';

        #* ++++++++++++++++++
        #*   debug settings

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #writeMailsToDiskDuringDebug = $true;

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #alwaysWriteMailsToDiskInsteadSending = $false;

        #* can be used to redirect all mails to the specified
        #*   account(s)
        #redirectAllMails = @();

        #* this parameter can be used to force client processes
        #*   (--exec) to send PowerShell output serialzied to the
        #*   main proces (--run) Default: $false
        #SendOutputToMainProcess = $false;

        #* this will dump RER file contents to the Log.
        #DumpRerEventDataToLog = $false

        #* It is possible to save processed remote event files.
        #*   Normally they are deleted. Saving can be enabled in the
        #*   global configuration but must be also enabled for each
        #*   workflow itself by setting this property to $TRUE.
```

(continues on next page)

(continued from previous page)

```
        #*   Default: $FALSE
        #SaveRerEventsAfterProcessing = $false
    }

    if(!$kenaflow){import-module "<<directory>>\kenaflow.runtime.dll";Test-
↪KFConfig;exit}
```

17.8.3 List Workflow Complete Template

```
    @{
        #* A workflow will only work with the following version of
        #*   kenaflow. After you tested your workflow with a new
        #*   version of kenaflow you can change the following
        #*   setting.
        Version = "4.0";

        #* set it to $true to ENABLE the workflow. ($false DISABLES it)
        Enabled = $false;

        #* Type = SHAREPOINT, EMAIL, POWERSHELL
        Type = "SHAREPOINT";

        #* SubType
        #*   if "SHAREPOINT" then ALL, LASTMODIFIED (alias LIST),
        #*   QUERY, SITE, STATEMACHINE if "EMAIL" then
        #*   EXCHANGE2007SP1, EXCHANGE2010, EXCHANGE2010SP1,
        #*   EXCHANGE2010SP2, EXCHANGE2013, EXCHANGE2013SP1,
        #*   EXCHANGE, EXCHANGEONLINE, POP3, IMAP if "POWERSHELL" => EMPTY
        SubType = "LIST";

        #* set it to a meaningful name
        Name = "<<wfname>>";

        #* what SharePoint do you use? Possible values: "sp2013" or
        #*   "sp2016" or "sp2019" or "spo"
        Platform = "";

        #* a unique (!) id (type 'System.Guid') for this workflow.
        #*   must be unique in YOUR environment. Use
        #*   "[guid]::newGuid().tostring('d')" to create a GUID in
        #*   Powershell.
        Id = "<<wfid>>";

        #* A list of one or more files that are checked during deserialization.
        #*   The workflow configuration is reserialized if such an additional
        #*   file was modified since last serialization or is missing.
        #AdditionalConfigFiles = @();

        #* option. can be used to specify the folder where the
        #*   workflow scripts are in. the workflow directory (where
        #*   the _wfconfig.ps1 is in) is used if empty.
        #ScriptFolder = "";

        #* "Time between execution" => minimal time difference IN
        #*   SECONDS between two executions of the workflow
        TBE = 60;

        #* CRON-style pattern for execution scheduling. Please
```

(continues on next page)

(continued from previous page)

```
        #*   consult our documentation to get more infos on that.
        #*   Make sure you set TBE > 0 to activate time-based
        #*   execution. First TBE is checked and afterwards CRON if
        #*   set.
        #Cron = "0 * * * *";

        #* During a time based workflow run the same workflow
        #*   should start also on another kenaflow server (in case of
        #*   scaling). If your really sure that you want to run
        #*   multiple instances of the same workflow than you can set
        #*   this to $FALSE.
        #Locking = $true;

        #* This can be used to specify one or more custom
        #*   PowerShell libariy files that will be executed before
        #*   workflow script execution.
        #CustomLibrary = @("");

        #* Here you can specify a script for error handling. It's
        #*   processed in case of exceptions within the workflow
        #*   script. Please read
        #*   https://doc.kenaflow.com/basics/errorhandling
        #ErrorHandlingScript = "errorHandling.ps1"

        #* Can be used to mark this workflow as "in debug" for use
        #*   with cmdlet Get-KFInDebug -WorkflowDebugState
        #Debug = $false;

        #* This file will be serialized for performance reasons.
        #*   You can prevent serialization by setting this option.
        #NoSerializing = $false;

        #* This file will be serialized for performance reasons as
        #*   long as property 'NoSerializing' is not set to $true.
        #*   But you can specify a "maximum lifetime" here. if this
        #*   amount of time elapsed the workflow configuration gets
        #*   re-serialized. Possible values are TimeSpan (object or
        #*   string) or amount of seconds (int or string)
        #MaxLifetimeOfSerialization = "0:5" #5 minutes

        #UserProfileCacheLifetime = 60; #In seconds. If not specified the global␣
↪setting from kenaflow.conf is used

        #* Force workflow script to run again in case of "version
        #*   conflict" (parallel item modifications in SharePoint).
        #*   Default: $false
        #RetryOnVersionConflict = $false;

        #* If set to $false the workflow will continue with the
        #*   next item on workflow script errors. Default ist $true
        #*   ("stop workflow on script errors")
        #StopWorkflowOnScriptError = $true;


        #* The following two parameters can be used to specify credentials
        #*   however it is possible to store credentials in a
        #*   protected fashion using the parameters --setusername
        #*   and --setpassword on kenaflow.exe while running the
        #*   program inside a workflow folder that contains
        #*   _wfconfig.ps1. this will create a file
        #*   _wfconfig-cred.ps1 that contains encrypted credentials.
```

(continues on next page)

(continued from previous page)

```
        #*   the credentials can only decrypted on the same machine.
        #*   on other machines the file is worthless.
        #spUser = "";
        #spPwd = '';

        #* only if platform is 'spo' = SharePoint Online
        #spoAppId = "";
        #* only if platform is 'spo' = SharePoint Online
        #spoAppSecret = '';

        #* You can execute a SharePoint
        #*   workflow on multiple lists in multiple webs. The
        #*   executed workflow script is always the same. All other
        #*   settings are identical (names of lists such as error
        #*   list or data list). But the lists can be configured per
        #*   web. If you configure this property you cannot
        #*   configure `Web`, `AlternateWebUrls` and `List` in
        #*   parallel.
        <#
        Connection = @(
          @{
            Web = "https://firstweb.sharepoint.farm";
            List = @("List 1", "List 2");
            AlternateWebUrls = @();
          },
          @{
            Web = "https://secondweb.sharepoint.farm";
            List = "List 3";
            AlternateWebUrls = @();
          },
          @{
            Web = "https://thirdweb.sharepoint.farm";
            List = @("List 4", "List 5");
            AlternateWebUrls = @();
          }
        )
        #>

        #* Full URL to the SharePoint web where the workflow list
        #*   is in
        Web = "";

        #* Title of the SharePoint list where the workflow operates
        #*   on. This list is called "workflow list". You can
        #*   configure multiple list for the workflow by using a
        #*   PowerShell list object: @("List 1", "List 2")
        List = @("");


        #* A list of alternate URLS, e.g. if a web application has
        #*   Alternate Access Mappings. Remove Events could be
        #*   received with such an alternate URL
        #AlternateWebUrls = @();

        #* a list for global config values
        ConfigList = "WFconfig";

        #* a list that can be used to store data
        DataList = "WFData";

        #* a list that can be used to record log information
```

(continues on next page)

(continued from previous page)

```
        HistoryList = "WFHistory";

        #* Workflow errors will be stored in this SharePoint list
        ErrorList = "WFErrors";

        #* Can be uses to define default config entries that will
        #*   be deployed to the config list if the workflow config
        #*   list is created by kenaflow (kenaflow.exe)
        ConfigListDefaults = @{
            "kenaflow" = "cool"
        };

        #* List of mail addresses that will receive mails in case
        #*   of workflow internal errors.
        AlertAddresses = @();

        #* can be used to overwrite the global settings.
        #AlertFloodProtection = 3600;

        #* Address of the sender of mails that the workflow sends.
        #*   If empty it will be taken from the kenaflow global
        #*   config
        #MailFrom = "";

        #* Address to which uses can reply to. If empty it will be
        #*   taken from the kenaflow global config
        #MailReply = "";

        #* This script contains all to workflow actions
        #Script = "script.ps1";

        #* After this amount of time elasped the workflow execution
        #*   will be stopped; -1 = use default
        #MaxExecutionTime = -1;

        #* This script is executed before querying
        #PreQueryProcessing = "preQueryProcessing.ps1";

        #* Specified whether the Workflow needs Remote Event
        #*   Receiver to execute the workflow scripts immediately
        #*   after a change in SharePoint.
        RER = $false;

        #* Optional, but must be used if multipe workflow listen to
        #*   the same list!!
        #RERHandlerName = "kenaflow_wf1";

        #* Specifies how long a Remote Event is valid before it is
        #*   skipped forever (if not processed). Default:
        #*   configuration in global config. There is default: 120s.
        #MaximumRerLifetime = 120;

        #* Specifies how long a a remote event is postponed in case
        #*   of errors during processing. Default: configuration in
        #*   global config. There is default: 5s.
        #PostponeFailedRer = 5;

        #* if false the 'last modified' flag file is used
        #LastModifiedBasedOnLastRun = true;

        #* Count of list items queried from SharePoint at one
```

(continues on next page)

(continued from previous page)

```
        #*   request. -1 = use default
        #ItemBatchCount = -1;

        #* Amount of time before each item is checked to be
        #*   processed. -1 = use default; 0 = check immediately
        #ItemReCheckTime = -1;

        #* Handle lists and libraries "flat" by ignoring folders.
        #*   Default: $false
        #IgnoreFoldersInQuery = $false;

        #* In case of using SubType 'QUERY' you can specify a query
        #*   for item processing here.
        #Query="{{Title}} == string 'Test' & {{Modified}} > datetime [[TODAY]] & {
↪{Editor}} != lookupid [[USERID]]"; #used only if 'Type' == 'query'
        #* You can specify CAML instead. {{ }} tokens are processed
        #*   at runtime
        #Query="<Where><And><And><Eq><FieldRef Name='Title'/>"+
        #      "<Value Type='Text'>Test</Value></Eq><Gt>"+
        #      "<FieldRef Name='Modified'/><Value Type='DateTime' IncludeTimeValue=
↪'TRUE'>"+
        #      "<Today/></Value></Gt></And>" +
        #      "<Neq><FieldRef Name='Editor'/><Value Type='Lookup'>"+
        #      "<UserId/></Value></Neq></And></Where>";

        #* If you only want to process item with changes in certain
        #*   item fields (columns) you can specify the list of names
        #*   here. If you want only process items with changes in any
        #*   field you can set this simply to $true.
        #ItemChangeHashFields = $true;
        #ItemChangeHashFields = @("Title", "CustomField");

        #* Some item permission related Cmdlets in workflows can
        #*   simply set "CRUD" +M (Manage) +A (Approve) permissions
        #*   by specifying some characters from set [C, R, U, D, M,
        #*   A]. They refer to permission set that the Site
        #*   Collection administrator has to create. They must also
        #*   be named here
        #PermissionSetCreate="";
        #PermissionSetRead="";
        #PermissionSetUpdate="";
        #PermissionSetDelete="";
        #PermissionSetManage="";
        #PermissionSetApprove="";


        #* ++++++++++++++++++
        #*   runtime settings

        #* (only) kenaflow Cmdlets will throw Exceptions on failure
        #*   ($true) or hide them ($false). In the last case the
        #*   exception can be queried with Get-KFLastResult. Default
        #*   is $true
        #ThrowErrorsPreference = $true;
        #* This is the default for the common PowerShell variable
        #*   $ErrorActionPreference that controls the default
        #*   PowerShell error handling. Default is "Stop". This can
        #*   be overwritten by Invoke-Kenaflow -ErrorActionPreference
        #*   <value> or in script using $ErrorActionPreference =
        #*   <value>
        #ErrorActionPreference = 'Stop';
```

(continues on next page)

(continued from previous page)

```
        #* ++++++++++++++++++
        #*   debug settings

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #writeMailsToDiskDuringDebug = $true;

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #alwaysWriteMailsToDiskInsteadSending = $false;

        #* can be used to redirect all mails to the specified
        #*   account(s)
        #redirectAllMails = @();

        #* this parameter can be used to force client processes
        #*   (--exec) to send PowerShell output serialzied to the
        #*   main proces (--run) Default: $false
        #SendOutputToMainProcess = $false;

        #* this will dump RER file contents to the Log.
        #DumpRerEventDataToLog = $false

        #* It is possible to save processed remote event files.
        #*   Normally they are deleted. Saving can be enabled in the
        #*   global configuration but must be also enabled for each
        #*   workflow itself by setting this property to $TRUE.
        #*   Default: $FALSE
        #SaveRerEventsAfterProcessing = $false
    }

    if(!$kenaflow){import-module "<<directory>>\kenaflow.runtime.dll";Test-
↪KFConfig;exit}
```

17.8.4 State Machine Workflow Complete Template

```
    @{
        #* A workflow will only work with the following version of
        #*   kenaflow. After you tested your workflow with a new
        #*   version of kenaflow you can change the following
        #*   setting.
        Version = "4.0";

        #* set it to $true to ENABLE the workflow. ($false DISABLES it)
        Enabled = $false;

        #* Type = SHAREPOINT, EMAIL, POWERSHELL
        Type = "SHAREPOINT";

        #* SubType
        #*   if "SHAREPOINT" then ALL, LASTMODIFIED (alias LIST),
        #*   QUERY, SITE, STATEMACHINE if "EMAIL" then
        #*   EXCHANGE2007SP1, EXCHANGE2010, EXCHANGE2010SP1,
        #*   EXCHANGE2010SP2, EXCHANGE2013, EXCHANGE2013SP1,
        #*   EXCHANGE, EXCHANGEONLINE, POP3, IMAP if "POWERSHELL" => EMPTY
        SubType = "STATEMACHINE";
```

(continues on next page)

(continued from previous page)

```
        #* set it to a meaningful name
        Name = "<<wfname>>";

        #* what SharePoint do you use? Possible values: "sp2013" or
        #*   "sp2016" or "sp2019" or "spo"
        Platform = "";

        #* a unique (!) id (type 'System.Guid') for this workflow.
        #*   must be unique in YOUR environment. Use
        #*   "[guid]::newGuid().tostring('d')" to create a GUID in
        #*   Powershell.
        Id = "<<wfid>>";

        #* A list of one or more files that are checked during deserialization.
        #*   The workflow configuration is reserialized if such an additional
        #*   file was modified since last serialization or is missing.
        #AdditionalConfigFiles = @();

        #* option. can be used to specify the folder where the
        #*   workflow scripts are in. the workflow directory (where
        #*   the _wfconfig.ps1 is in) is used if empty.
        #ScriptFolder = "";

        #* "Time between execution" => minimal time difference IN
        #*   SECONDS between two executions of the workflow Make sure
        #*   you set TBE > 0 to activate time-based execution. First
        #*   TBE is checked and afterwards CRON if set.
        TBE = 60;

        #* CRON-style pattern for execution scheduling. Please
        #*   consult our documentation to get more infos on that.
        #Cron = "0 * * * *";

        #* During a time based workflow run the same workflow
        #*   should start also on another kenaflow server (in case of
        #*   scaling). If your really sure that you want to run
        #*   multiple instances of the same workflow than you can set
        #*   this to $FALSE.
        #Locking = $true;

        #* This can be used to specify one or more custom
        #*   PowerShell libariy files that will be executed before
        #*   workflow script execution.
        #CustomLibrary = @("");

        #* Here you can specify a script for error handling. It's
        #*   processed in case of exceptions within the workflow
        #*   script. Please read
        #*   https://doc.kenaflow.com/basics/errorhandling
        #ErrorHandlingScript = "errorHandling.ps1"

        #* Can be used to mark this workflow as "in debug" for use
        #*   with cmdlet Get-KFInDebug -WorkflowDebugState
        #Debug = $false;

        #* This file will be serialized for performance reasons.
        #*   You can prevent serialization by setting this option.
        #NoSerializing = $false;

        #* This file will be serialized for performance reasons as
        #*   long as property 'NoSerializing' is not set to $true.
```

(continues on next page)

(continued from previous page)

```
        #*   But you can specify a "maximum lifetime" here. if this
        #*   amount of time elapsed the workflow configuration gets
        #*   re-serialized. Possible values are TimeSpan (object or
        #*   string) or amount of seconds (int or string)
        #MaxLifetimeOfSerialization = "0:5" #5 minutes

        #UserProfileCacheLifetime = 60; #In seconds. If not specified the global␣
↪setting from kenaflow.conf is used

        #* Force workflow script to run again in case of "version
        #*   conflict" (parallel item modifications in SharePoint).
        #*   Default: $false
        #RetryOnVersionConflict = $false;

        #* If set to $false the workflow will continue with the
        #*   next item on workflow script errors. Default ist $true
        #*   ("stop workflow on script errors")
        #StopWorkflowOnScriptError = $true;


        #* The following two parameters can be used to specify credentials
        #*   however it is possible to store credentials in a
        #*   protected fashion using the parameters --setusername
        #*   and --setpassword on kenaflow.exe while running the
        #*   program inside a workflow folder that contains
        #*   _wfconfig.ps1. this will create a file
        #*   _wfconfig-cred.ps1 that contains encrypted credentials.
        #*   the credentials can only decrypted on the same machine.
        #*   on other machines the file is worthless.
        #spUser = "";
        #spPwd = '';

        #* You can execute a SharePoint
        #*   workflow on multiple webs (and multiple lists if
        #*   workflow types *list* and *state machine*). The executed
        #*   workflow script is always the same. All other settings
        #*   are identical (names of lists such as error list or data
        #*   list). If you configure this property you cannot
        #*   configure `Web` and `AlternateWebUrls` in parallel.

        #* only if platform is 'spo' = SharePoint Online
        #spoAppId = "";
        #* only if platform is 'spo' = SharePoint Online
        #spoAppSecret = '';

        #* You can execute a SharePoint
        #*   workflow on multiple lists in multiple webs. The
        #*   executed workflow script is always the same. All other
        #*   settings are identical (names of lists such as error
        #*   list or data list). But the lists can be configured per
        #*   web. If you configure this property you cannot
        #*   configure `Web`, `AlternateWebUrls` and `List` in
        #*   parallel.
        <#
        Connection = @(
          @{
            Web = "https://firstweb.sharepoint.farm";
            List = @("List 1", "List 2");
            AlternateWebUrls = @();
          },
          @{
```

(continues on next page)

(continued from previous page)

```
    Web = "https://secondweb.sharepoint.farm";
    List = "List 3";
    AlternateWebUrls = @();
  },
  @{
    Web = "https://thirdweb.sharepoint.farm";
    List = @("List 4", "List 5");
    AlternateWebUrls = @();
  }
)
#>

#* Full URL to the SharePoint web where the workflow list
#*   is in
Web = "";

#* Title of the SharePoint list where the workflow operates
#*   on. This list is called "workflow list" You can
#*   configure multiple list for the workflow by using a
#*   PowerShell list object: @("List 1", "List 2")
List = @("");


#* A list of alternate URLS, e.g. if a web application has
#*   Alternate Access Mappings. Remove Events could be
#*   received with such an alternate URL
#AlternateWebUrls = @();

#* a list for global config values
ConfigList = "WFconfig";

#* a list that can be used to store data
DataList = "WFData";

#* a list that can be used to record log information
HistoryList = "WFHistory";

#* Workflow errors will be stored in this SharePoint list
ErrorList = "WFErrors";

#* Can be uses to define default config entries that will
#*   be deployed to the config list if the workflow config
#*   list is created by kenaflow (kenaflow.exe)
ConfigListDefaults = @{
    "kenaflow" = "cool"
};

#* List of mail addresses that will receive mails in case
#*   of workflow internal errors.
AlertAddresses = @();

#* can be used to overwrite the global settings.
#AlertFloodProtection = 3600;

#* Address of the sender of mails that the workflow sends.
#*   If empty it will be taken from the kenaflow global
#*   config
#MailFrom = "";

#* Address to which uses can reply to. If empty it will be
#*   taken from the kenaflow global config
```

(continues on next page)

(continued from previous page)

```
        #MailReply = "";

        #* After this amount of time elasped the workflow execution
        #*   will be stopped; -1 = use default
        #MaxExecutionTime = -1;

        #* This script is executed before querying
        #PreQueryProcessing = "preQueryProcessing.ps1";

        #* Specified whether the Workflow needs Remote Event
        #*   Receiver to execute the workflow scripts immediately
        #*   after a change in SharePoint.
        RER = $false;

        #* Optional, but must be used if multipe workflow listen to
        #*   the same list!!
        #RERHandlerName = "kenaflow_wf1";

        #* Specifies how long a Remote Event is valid before it is
        #*   skipped forever (if not processed). Default:
        #*   configuration in global config. There is default: 120s.
        #MaximumRerLifetime = 120;

        #* Specifies how long a a remote event is postponed in case
        #*   of errors during processing. Default: configuration in
        #*   global config. There is default: 5s.
        #PostponeFailedRer = 5;

        #* Handle lists and libraries "flat" by ignoring folders.
        #*   Default: $false
        #IgnoreFoldersInQuery = $false;

        #* lookup field in the workflow list that represents the
        #*   current item state
        StateField = "State";

        #* list that holds all possible states
        StateList = "WFStates";

        #* If you only want to process item with changes in certain
        #*   item fields (columns) you can specify the list of names
        #*   here. If you want only process items with changes in any
        #*   field you can set this simply to $true. This global
        #*   setting is used also for states if it is not overwritten
        #*   for a certain state!
        #ItemChangeHashFields = $true;
        #ItemChangeHashFields = @("Title", "CustomField");

        #* if false the 'last modified' flag file is used
        #LastModifiedBasedOnLastRun = true;

        #* Count of list items queried from SharePoint at one
        #*   request. -1 = use default
        #ItemBatchCount = -1;

        #* Amount of time before each item is checked to be
        #*   processed. -1 = use default; 0 = check immediately
        #ItemReCheckTime = -1;

        States = @{
            "(empty)" = @{
```

(continues on next page)

(continued from previous page)

```
                #* You can define the execution order for states by
                #*   defining this key. All states that have not
                #*   definied this setting the number 0 is used.
                #*   Except for state "(empty)" for which -1 is used
                #*   by default. So (empty) will be executed first.
                #*   All scripts with the same "ExecutionOrder" (or
                #*   none which means "0") are executed by the order
                #*   of their keys.
                #ExecutionOrder = 100;

                #* After this run time SharePoint list items will
                #*   be checked for their corrent state before
                #*   processing
                #ItemReCheckTime = -1;

                #* After this amount of time elasped the workflow
                #*   execution will be stopped; -1 = use default
                #MaxExecutionTime = -1;

                #* Count of list items queried from SharePoint at
                #*   one request. -1 = use default
                #ItemBatchCount = -1;

                #* Can be used to mark this state as "in debug" for
                #*   use with cmdlet Get-KFInDebug
                #*   -WorkflowStateDebugState
                #Debug = $false;

                #LastModifiedBasedOnLastRun = true; #if false the 'last modified'␣
↪flag file is used
            };
            "1.0" = @{
                #* States can be disabled for processing by setting
                #*   this to $false. Default is $true.
                Enabled = $true;

                #* if not set the State key ("1.0") will be used
                Title = "State 1.0";

                #* query, lastmodified, all or (empty)
                Type="lastmodified";

                #* will be translated into CAML
                #Query = "";

                #* If you only want to process item with changes in
                #*   certain item fields (columns) you can specify
                #*   the list of names here. If you want only process
                #*   items with changes in any field you can set this
                #*   simply to $true. If you want to disable this of
                #*   the current state and the global setting of the
                #*   workflow for this property defines a field list
                #*   or $true, then you can set this to $false
                #ItemChangeHashFields = $true;
                #ItemChangeHashFields = $false;
                #ItemChangeHashFields = @("Title", "CustomField");

                #* Handle lists and libraries "flat" by ignoring
                #*   folders. Default: $false
                #IgnoreFoldersInQuery = $false;
```

(continues on next page)

(continued from previous page)

```
                #* if empty the script must be named as
                #*   "S_<state><your text or noting>.ps1"
                #Script = "";

                #* Time Before Execution. Each state remembers the
                #*   last runtime in a text file. The state script
                #*   will not execute until this amount of seconds
                #*   elapsed sind last run.
                #TBE = 30;

                #* CRON-style pattern for execution scheduling.
                #*   Please consult our documentation to get more
                #*   infos on that.
                #Cron = "0 * * * *";
            };
            "2.0" = @{
                Type="query";
                Query="{{Title}} == string 'Test' & {{Modified}} > datetime␣
↪[[TODAY]] & {{Editor}} != lookupid [[USERID]]"
                #You can specify CAML instead. {{ }} tokens are processed at␣
↪runtime
                #Query="<Where><And><And><Eq><FieldRef Name='Title'/><Value Type=
↪'Text'>Test</Value></Eq><Gt><FieldRef Name='Modified'/><Value Type='DateTime'␣
↪IncludeTimeValue='TRUE'><Today/></Value></Gt></And>" +
                #* "<Neq><FieldRef Name='Editor'/><Value Type='Lookup'><UserId/></
↪Value></Neq></And></Where>";

            };
            "3.0" = @{
                Title = "Last State";

                #* This state has a dedicated script name that is
                #*   used instead the default state script name
                #*   "S_<state-key>.ps1"
                Script = "S_laststate.ps1";
            };
            "4.0" = @{
                Enabled = $false;
                Title = "Ignore";
            };
        }


        #* Some item permission related Cmdlets in workflows can
        #*   simply set "CRUD" +M (Manage) +A (Approve) permissions
        #*   by specifying some characters from set [C, R, U, D, M,
        #*   A]. They refer to permission set that the Site
        #*   Collection administrator has to create. They must also
        #*   be named here
        #PermissionSetCreate="";
        #PermissionSetRead="";
        #PermissionSetUpdate="";
        #PermissionSetDelete="";
        #PermissionSetManage="";
        #PermissionSetApprove="";


        #* ++++++++++++++++++
        #*   runtime settings

        #* (only) kenaflow Cmdlets will throw Exceptions on failure
```

(continues on next page)

(continued from previous page)

```
        #*   ($true) or hide them ($false). In the last case the
        #*   exception can be queried with Get-KFLastResult. Default
        #*   is $true
        #ThrowErrorsPreference = $true;
        #* This is the default for the common PowerShell variable
        #*   $ErrorActionPreference that controls the default
        #*   PowerShell error handling. Default is "Stop". This can
        #*   be overwritten by Invoke-Kenaflow -ErrorActionPreference
        #*   <value> or in script using $ErrorActionPreference =
        #*   <value>
        #ErrorActionPreference = 'Stop';

        #* ++++++++++++++++++
        #*   debug settings

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #writeMailsToDiskDuringDebug = $true;

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #alwaysWriteMailsToDiskInsteadSending = $false;

        #* can be used to redirect all mails to the specified
        #*   account(s)
        #redirectAllMails = @();

        #* this parameter can be used to force client processes
        #*   (--exec) to send PowerShell output serialzied to the
        #*   main proces (--run) Default: $false
        #SendOutputToMainProcess = $false;

        #* this will dump RER file contents to the Log.
        #DumpRerEventDataToLog = $false

        #* It is possible to save processed remote event files.
        #*   Normally they are deleted. Saving can be enabled in the
        #*   global configuration but must be also enabled for each
        #*   workflow itself by setting this property to $TRUE.
        #*   Default: $FALSE
        #SaveRerEventsAfterProcessing = $false
    }

    if(!$kenaflow){import-module "<<directory>>\kenaflow.runtime.dll";Test-
↪KFConfig;exit}
```

17.8.5 Email Workflow Complete Template

```
    @{
        #* A workflow will only work with the following version of
        #*   kenaflow. After you tested your workflow with a new
        #*   version of kenaflow you can change the following
        #*   setting.
        Version = "4.0";

        #* set it to $true to ENABLE the workflow. ($false DISABLES it)
        Enabled = $false;
```

(continues on next page)

(continued from previous page)

```
        #* Type = SHAREPOINT, EMAIL, PowerShell
        Type = "EMAIL";

        #* SubType
        #*   if "SHAREPOINT" then ALL, LASTMODIFIED (alias LIST),
        #*   QUERY, SITE, STATEMACHINE if "EMAIL" then
        #*   EXCHANGE2007SP1, EXCHANGE2010, EXCHANGE2010SP1,
        #*   EXCHANGE2010SP2, EXCHANGE2013, EXCHANGE2013SP1,
        #*   EXCHANGE, EXCHANGEONLINE, POP3, IMAP if "POWERSHELL" => EMPTY
        SubType = "IMAP";

        #* set it to a meaningful name
        Name = "<<wfname>>";

        #* a unique (!) id (type 'System.Guid') for this workflow.
        #*   must be unique in YOUR environment. Use
        #*   "[guid]::newGuid().tostring('d')" to create a GUID in
        #*   Powershell.
        Id = "<<wfid>>";

        #* A list of one or more files that are checked during deserialization.
        #*   The workflow configuration is reserialized if such an additional
        #*   file was modified since last serialization or is missing.
        #AdditionalConfigFiles = @();

        #* option. can be used to specify the folder where the
        #*   workflow scripts are in. the workflow directory (where
        #*   the _wfconfig.ps1 is in) is used if empty.
        #ScriptFolder = "";

        #* "Time between execution" => minimal time difference IN
        #*   SECONDS between two executions of the workflow
        TBE = 60;

        #* CRON-style pattern for execution scheduling. Please
        #*   consult our documentation to get more infos on that.
        #*   Make sure you set TBE > 0 to activate time-based
        #*   execution. First TBE is checked and afterwards CRON if
        #*   set.
        #Cron = "0 * * * *";

        #* During a time based workflow run the same workflow
        #*   should start also on another kenaflow server (in case of
        #*   scaling). If your really sure that you want to run
        #*   multiple instances of the same workflow than you can set
        #*   this to $FALSE.
        #Locking = $true;

        #* This can be used to specify one or more custom
        #*   PowerShell libariy files that will be executed before
        #*   workflow script execution.
        #CustomLibrary = @("");

        #* Here you can specify a script for error handling. It's
        #*   processed in case of exceptions within the workflow
        #*   script. Please read
        #*   https://doc.kenaflow.com/basics/errorhandling
        #ErrorHandlingScript = "errorHandling.ps1"

        #* Can be used to mark this workflow as "in debug" for use
        #*   with cmdlet Get-KFInDebug -WorkflowDebugState
```

(continues on next page)

(continued from previous page)

```
        #Debug = $false;

        #* This file will be serialized for performance reasons.
        #*   You can prevent serialization by setting this option.
        #NoSerializing = $false;

        #* This file will be serialized for performance reasons as
        #*   long as property 'NoSerializing' is not set to $true.
        #*   But you can specify a "maximum lifetime" here. if this
        #*   amount of time elapsed the workflow configuration gets
        #*   re-serialized. Possible values are TimeSpan (object or
        #*   string) or amount of seconds (int or string)
        #MaxLifetimeOfSerialization = "0:5" #5 minutes

        #* If set to $false the workflow will continue with the
        #*   next item on workflow script errors. Default ist $true
        #*   ("stop workflow on script errors")
        #StopWorkflowOnScriptError = $true;

        #* For Exchange (on-premises or Online)
        #EmailDomain = "example.com";

        #* E-Mail Account
        EmailUsername = "";

        #* E-Mail Password
        EmailPassword = "";

        #* Ignore SSL cert errors ?
        #*   $true (default) or $false
        EmailIgnoreSSLErrors = $true;

        #* E-Mail address of the monitored account - e.g.
        #*   kenaflow@example.com - to auto-discover the Exchange
        #*   server Required if SUBTYPE is EXCHANGE*
        #EmailExchangeAutodiscoverEmailAddress = "";

        #* URL of the Exchange Service
        #EmailExchangeServiceUrl = "";

        #* Server addess (required for SUBTYPE == POP3 or == IMAP)
        #EmailServerAddress = "imap.example.com";

        #* Security Settings (for IMAP and POP3) => AUTO (default),
        #*   NONE, SSLONCONNECT, STARTTLS, STARTTLSWHENAVAILABLE
        #EmailServerSecurity = "AUTO";

        #* Port of the mail server (for IMAP and POP3); -1 is
        #*   default: guess 'port' from 'Subtype' (IMAP, POP3) and
        #*   'EmailServerSecurity'
        #EmailServerPort = -1;

    <#
    #*␣
↪===========================================================================================
    #*   when using E-Mail Workflow with SharePoint you need to
    #*   configure this section

        #* what SharePoint do you use? Possible values: "sp2013" or
        #*   "sp2016" or "sp2019" or "spo"
        Platform = "";
```

(continues on next page)

(continued from previous page)

```
        #* The following two parameters can be used to specify credentials
        #*   however it is possible to store credentials in a
        #*   protected fashion using the parameters --setusername
        #*   and --setpassword on kenaflow.exe while running the
        #*   program inside a workflow folder that contains
        #*   _wfconfig.ps1. this will create a file
        #*   _wfconfig-cred.ps1 that contains encrypted credentials.
        #*   the credentials can only decrypted on the same machine.
        #*   on other machines the file is worthless.
        #spUser = "";
        #spPwd = '';

        #* only if platform is 'spo' = SharePoint Online
        #spoAppId = "";
        #* only if platform is 'spo' = SharePoint Online
        #spoAppSecret = '';


        #* Full URL to the SharePoint web where the workflow list
        #*   is in
        Web = "";

        #* A list of alternate URLS, e.g. if a web application has
        #*   Alternate Access Mappings. Remove Events could be
        #*   received with such an alternate URL
        #AlternateWebUrls = @();

        #* a list for global config values
        ConfigList = "WFconfig";

        #* a list that can be used to store data
        DataList = "WFData";

        #* a list that can be used to record log information
        HistoryList = "WFHistory";

        #* Workflow errors will be stored in this SharePoint list
        ErrorList = "WFErrors";

        #* Force workflow script to run again in case of "version
        #*   conflict" (parallel item modifications in SharePoint).
        #*   Default: $false
        #RetryOnVersionConflict = $false;

        #UserProfileCacheLifetime = 60; #In seconds. If not specified the global␣
↪setting from kenaflow.conf is used


    #*␣
↪==========================================================================================
↪
    #*
    #>

        #* Can be uses to define default config entries that will
        #*   be deployed to the config list if the workflow config
        #*   list is created by kenaflow (kenaflow.exe)
        ConfigListDefaults = @{
            "kenaflow" = "cool"
        };
```

(continues on next page)

(continued from previous page)

```
    #* List of mail addresses that will receive mails in case
    #*   of workflow internal errors.
    AlertAddresses = @();
    #* can be used to overwrite the global settings.
    #AlertFloodProtection = 3600;
    #* Address of the sender of mails that the workflow sends.
    #*   If empty it will be taken from the kenaflow global
    #*   config
    #MailFrom = "";

    #* Address to which uses can reply to. If empty it will be
    #*   taken from the kenaflow global config
    #MailReply = "";

    #* This script contains all to workflow actions
    #Script = "script.ps1";

    #* After this amount of time elasped the workflow execution
    #*   will be stopped; -1 = use default
    #MaxExecutionTime = -1;



    #* ++++++++++++++++++
    #*   runtime settings

    #* (only) kenaflow Cmdlets will throw Exceptions on failure
    #*   ($true) or hide them ($false). In the last case the
    #*   exception can be queried with Get-KFLastResult. Default
    #*   is $true
    #ThrowErrorsPreference = $true;
    #* This is the default for the common PowerShell variable
    #*   $ErrorActionPreference that controls the default
    #*   PowerShell error handling. Default is "Stop". This can
    #*   be overwritten by Invoke-Kenaflow -ErrorActionPreference
    #*   <value> or in script using $ErrorActionPreference =
    #*   <value>
    #ErrorActionPreference = 'Stop';

    #* ++++++++++++++++++
    #*   debug settings

    #* if not specified here the default setting from kenaflow
    #*   will be used.
    #writeMailsToDiskDuringDebug = $true;

    #* if not specified here the default setting from kenaflow
    #*   will be used.
    #alwaysWriteMailsToDiskInsteadSending = $false;

    #* can be used to redirect all mails to the specified
    #*   account(s)
    #redirectAllMails = @();

    #* this parameter can be used to force client processes
    #*   (--exec) to send PowerShell output serialzied to the
    #*   main proces (--run) Default: $false
    #SendOutputToMainProcess = $false;
}
```

(continues on next page)

(continued from previous page)

```
    if(!$kenaflow){import-module "<<directory>>\kenaflow.runtime.dll";Test-
↪KFConfig;exit}
```

17.8.6 PowerShell Workflow Complete Template

```
    @{
        #* A workflow will only work with the following version of
        #*   kenaflow. After you tested your workflow with a new
        #*   version of kenaflow you can change the following
        #*   setting.
        Version = "4.0";

        #* set it to $true to ENABLE the workflow. ($false DISABLES it)
        Enabled = $false;

        #* Type = SHAREPOINT, EMAIL, PowerShell
        Type = "POWERSHELL";

        #* set it to a meaningful name
        Name = "<<wfname>>";

        #* a unique (!) id (type 'System.Guid') for this workflow.
        #*   must be unique in YOUR environment. Use
        #*   "[guid]::newGuid().tostring('d')" to create a GUID in
        #*   Powershell.
        Id = "<<wfid>>";

        #* A list of one or more files that are checked during deserialization.
        #*   The workflow configuration is reserialized if such an additional
        #*   file was modified since last serialization or is missing.
        #AdditionalConfigFiles = @();

        #* option. can be used to specify the folder where the
        #*   workflow scripts are in. the workflow directory (where
        #*   the _wfconfig.ps1 is in) is used if empty.
        #ScriptFolder = "";

        #* "Time between execution" => minimal time difference IN
        #*   SECONDS between two executions of the workflow Make sure
        #*   you set TBE > 0 to activate time-based execution. First
        #*   TBE is checked and afterwards CRON if set.
        TBE = 60;

        #* CRON-style pattern for execution scheduling. Please
        #*   consult our documentation to get more infos on that.
        #Cron = "0 * * * *";

        #* During a time based workflow run the same workflow
        #*   should start also on another kenaflow server (in case of
        #*   scaling). If your really sure that you want to run
        #*   multiple instances of the same workflow than you can set
        #*   this to $FALSE.
        #Locking = $true;

        #* This can be used to specify one or more custom
        #*   PowerShell libariy files that will be executed before
        #*   workflow script execution.
        #CustomLibrary = @("");
```

(continues on next page)

(continued from previous page)

```
    #* Here you can specify a script for error handling. It's
    #*   processed in case of exceptions within the workflow
    #*   script. Please read
    #*   https://doc.kenaflow.com/basics/errorhandling
    #ErrorHandlingScript = "errorHandling.ps1"

    #* Can be used to mark this workflow as "in debug" for use
    #*   with cmdlet Get-KFInDebug -WorkflowDebugState
    #Debug = $false;

    #* This file will be serialized for performance reasons.
    #*   You can prevent serialization by setting this option.
    #NoSerializing = $false;

    #* This file will be serialized for performance reasons as
    #*   long as property 'NoSerializing' is not set to $true.
    #*   But you can specify a "maximum lifetime" here. if this
    #*   amount of time elapsed the workflow configuration gets
    #*   re-serialized. Possible values are TimeSpan (object or
    #*   string) or amount of seconds (int or string)
    #MaxLifetimeOfSerialization = "0:5" #5 minutes

    #* If set to $false the workflow will continue with the
    #*   next item on workflow script errors. Default ist $true
    #*   ("stop workflow on script errors")
    #StopWorkflowOnScriptError = $true;

    #* Can be uses to define default config entries that will
    #*   be deployed to the config list if the workflow config
    #*   list is created by kenaflow (kenaflow.exe)
    ConfigListDefaults = @{
        "kenaflow" = "cool"
    };

    #* List of mail addresses that will receive mails in case
    #*   of workflow internal errors.
    AlertAddresses = @();

    #* can be used to overwrite the global settings.
    #AlertFloodProtection = 3600;

    #* Address of the sender of mails that the workflow sends.
    #*   If empty it will be taken from the kenaflow global
    #*   config
    #MailFrom = "";

    #* Address to which uses can reply to. If empty it will be
    #*   taken from the kenaflow global config
    #MailReply = "";

    #* This script contains all to workflow actions
    #Script = "script.ps1";

    #* After this amount of time elasped the workflow execution
    #*   will be stopped; -1 = use default
    #MaxExecutionTime = -1;

    #* Specified whether the Workflow needs Remote Event
    #*   Receiver to execute the workflow scripts immediately
    #*   after a change in SharePoint.
```

(continues on next page)

(continued from previous page)

```
        RER = $false;

        #* Specifies how long a Remote Event is valid before it is
        #*   skipped forever (if not processed). Default:
        #*   configuration in global config. There is default: 120s.
        #MaximumRerLifetime = 120;

        #* Specifies how long a a remote event is postponed in case
        #*   of errors during processing. Default: configuration in
        #*   global config. There is default: 5s.
        #PostponeFailedRer = 5;


        #* ++++++++++++++++++
        #*   runtime settings

        #* (only) kenaflow Cmdlets will throw Exceptions on failure
        #*   ($true) or hide them ($false). In the last case the
        #*   exception can be queried with Get-KFLastResult. Default
        #*   is $true
        #ThrowErrorsPreference = $true;
        #* This is the default for the common PowerShell variable
        #*   $ErrorActionPreference that controls the default
        #*   PowerShell error handling. Default is "Stop". This can
        #*   be overwritten by Invoke-Kenaflow -ErrorActionPreference
        #*   <value> or in script using $ErrorActionPreference =
        #*   <value>
        #ErrorActionPreference = 'Stop';

        #* ++++++++++++++++++
        #*   debug settings

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #writeMailsToDiskDuringDebug = $true;

        #* if not specified here the default setting from kenaflow
        #*   will be used.
        #alwaysWriteMailsToDiskInsteadSending = $false;

        #* can be used to redirect all mails to the specified
        #*   account(s)
        #redirectAllMails = @();

        #* this parameter can be used to force client processes
        #*   (--exec) to send PowerShell output serialzied to the
        #*   main proces (--run) Default: $false
        #SendOutputToMainProcess = $false;

        #* this will dump RER file contents to the Log.
        #DumpRerEventDataToLog = $false

        #* It is possible to save processed remote event files.
        #*   Normally they are deleted. Saving can be enabled in the
        #*   global configuration but must be also enabled for each
        #*   workflow itself by setting this property to $TRUE.
        #*   Default: $FALSE
        #SaveRerEventsAfterProcessing = $false
    }

    if(!$kenaflow){import-module "<<directory>>\kenaflow.runtime.dll";Test-
↪KFConfig;exit}
```

(continues on next page)

(continued from previous page)

CHAPTER

EIGHTEEN

HOW TO

In this section we describe common topics.

1. How to create app credentials in SharePoint Online / On-Premises
2. How to use GIT to manage different developmen/production stages of a workflow
3. Here we describe a common error when using Invoke-WebRequest from within a workflow.
4. Some comments about SharePoint list item moderation state.

18.1 App ID & App Secret

To use the workflow engine with SharePoint Online in the O365 cloud it is possible to use a normal account for accessing SharePoint.

But it is also possible to use “App Credentials”. In this section we describe how to use them.

This is important so that the credentials in the workflow configurations are not personal credentials.

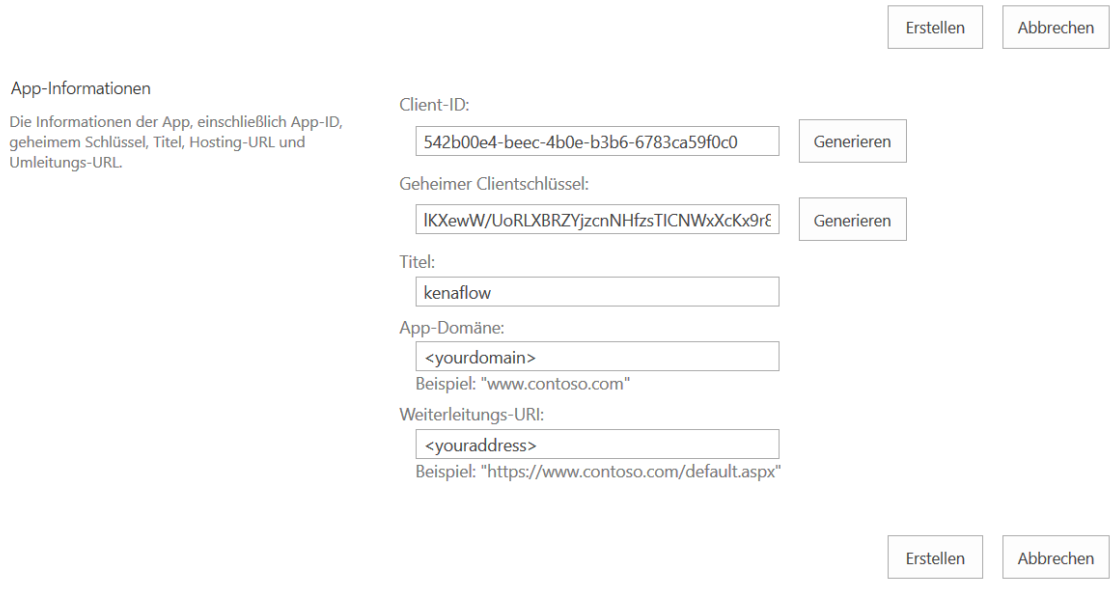
1. Log on to the SiteCollection that you want the Worklfow to access.

```
https://<your>.sharepoint.com/sites/<collection>
```

2. Navigate to the App registration form

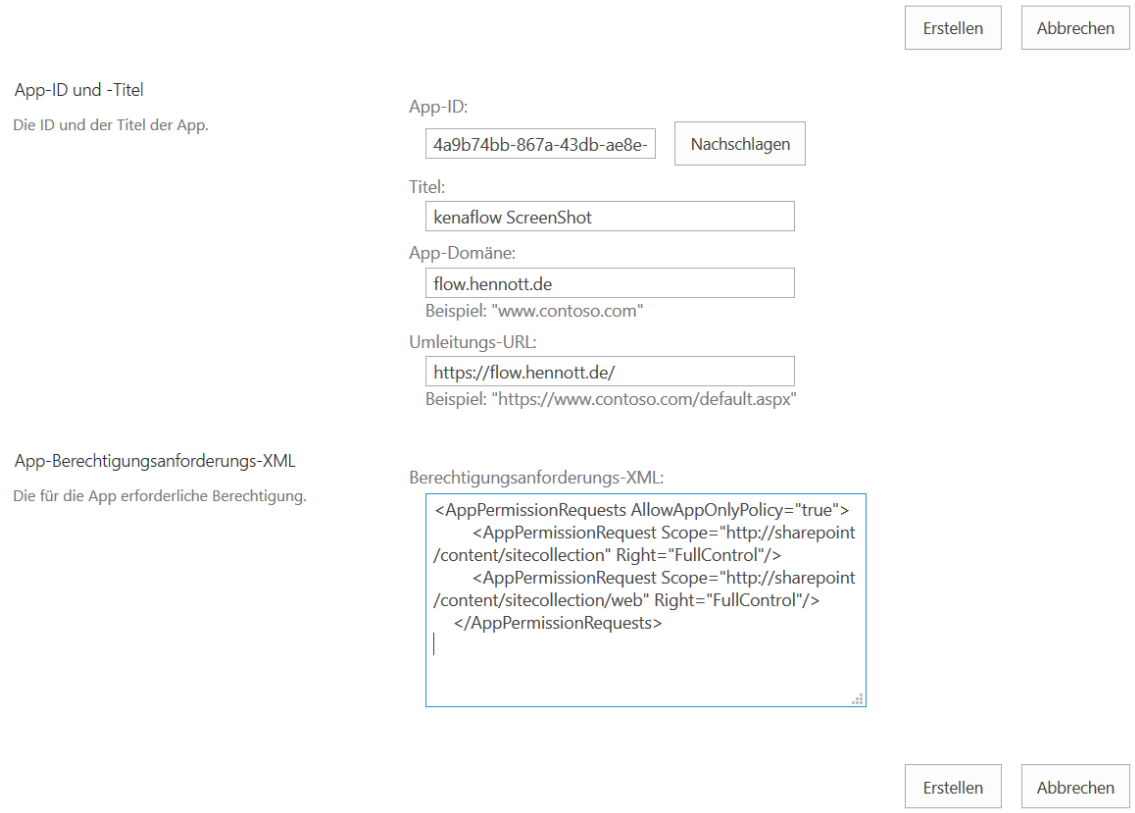
```
https://<your>.sharepoint.com/sites/<collection>/_layouts/AppRegNew.aspx
```

3. Create a new AppID and AppSecret



4. Now you have to assign permissions to the new key

```
https://<your>.sharepoint.com/sites/<collection>/_layouts/AppInv.aspx
```

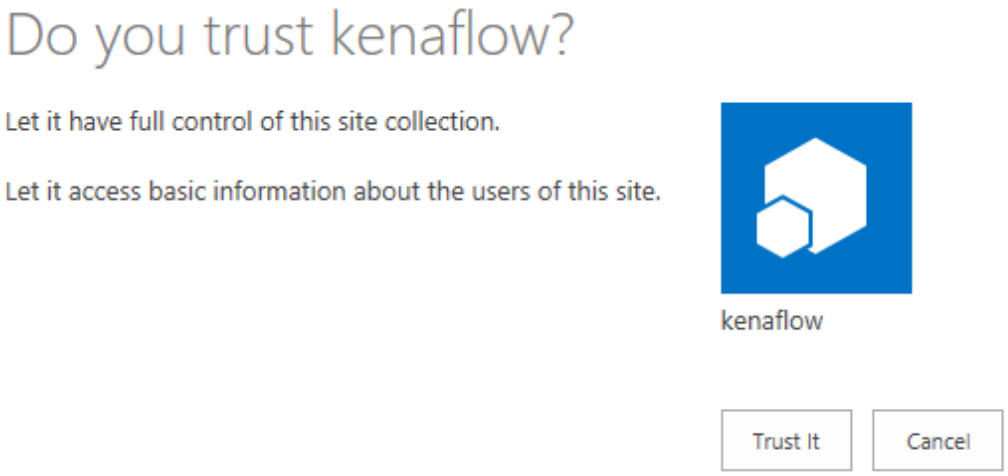


- Use the ClientID as AppId
- Insert the following XML to grant FullControl permissions to the key
- Please make sure that the xml is correctly formatted, otherwise the authorizations will not be correctly assigned

```
    <AppPermissionRequests AllowAppOnlyPolicy="true">
        <AppPermissionRequest Scope="http://sharepoint/content/
↪sitecollection" Right="FullControl"/>
        <AppPermissionRequest Scope="http://sharepoint/content/
↪sitecollection/web" Right="FullControl"/>
    </AppPermissionRequests>
```

- For more information about the xml https://dev.office.com/sharepoint/docs/sp-add-ins/add-in-permissions-in-sharepoint

5. If everything has been entered correctly, the result will look like this



6. Enter the AppID and AppSecret at **_wfconfig.ps1**

```
#* only if platform is 'spo' ╞ SharePoint Online
#spoAppId = "";
spoAppId="4a9b74bb-867a-43db-ae8e-2df9ed927d0e";

#* only if platform is 'spo' = SharePoint Online
#spoAppSecret = '';
spoAppSecret = "sEkubKOTpaGmXVrlY2NAwMXuzDKR4Z/w94mXz6aXlaI=";
```

18.1.1 Delete AppCredentials

You can revoke and delete the AppCredentials at any time. All you have to do is remove them from the SiteCollection using the following link.

```
https://<your>.sharepoint.com/sites/<collection>/_layouts/15/appprincipals.aspx
```

18.2 Using GIT in Workflow Development

18.2.1 Introduction

It is always a good idea to first test your software, or in this case workflows, before you integrate them into your live-system. Not doing this can result in malfunctioning workflows, in some cases even permanent loss of data or other irreparable damage to your system. In the following section we will give you a nice and easy way to develop workflows in the different stages of development (i.e. development, test-phase1, test-phase2, production), synchronize them and cheery-pick selected lines from one stage to the other, via Git. Git is a free software, for almost all plattforms (Windows, MacOS, Linux etc.), which was developped for excactly this purpose. In this little showcase we used Git Extensions the downloadable Git GUI. It is not necessary to download and use the Git Extension. Everything that is going to be shown in the following sections could be just as well done via Git in the command prompt.

18.2.2 Showcase

The goal of this showcase is to show in detail how to create an architecture that enables you to create and manage different stages of development for your workflows. The following steps are going to be described in detail after some background information about some aspects of Git.

1. Create a development directory
2. Create its contents (wfconfing & script)
3. Initialise Git
4. Commit the contents of your directory to Git
5. Create a branch
6. Create a repository directory
7. Add the repository to the development directory
8. Push the contents of the development directory
9. Copy of the development directory
10. Rename and change contents of the copy
11. Create branch of the renamed copy
12. Push the contents of the copy
13. Make a change in the development directory
14. Commit and Push those changes
15. Pull those changes from the renamed copy branch
16. Cherry pick the changes you want
17. Commit and Push those changes
18. Pull (development)

18.2.3 Creating a Git repository

A repository exisits in two forms in Git:

1. Personal repository
2. Central repository

The personal repository is selected for every one of the desired devolpment stages to later link them to a central repository. The central repository functions as the administrator of all the personal repositories. It is the place where the data of the personal repositories is collected.

The first step is creating a folder for the first stage of development. After that you need to create the content of your workflow which consits of at least the workflow-config and one script. At this point you want to initialise this folder as a Git personal repository. To do this right-click at an empty space of the choosen folder and select "GitExt Create new repository…". The first field is the directory, which in the most cases can be left as it is. The second field is the option of the presonal and central repository. Choose personal, since this is your folder for the first stage of development.

The next steps are committing the contents of your development folder via Git and creating a branch. A commit is a snapshot of the current state your data is in a the moment

1. Commit your files by right-clicking in the development folder and click on "GitExt Commit…" .
2. Select all the files you want to comming and enter a "Commit message".
3. Click on the commit button.

Now the data is commited via Git.

After committing the data you need to create a branch. A branch is a pointer to a commit. Each stage of development should have their own branch. To create a branch follow these steps.

1. Open "Git Browse" by right-clicking somewhere in the development folder.
2. Right-click on the first entry and select create new branch.
3. Name the branch in a way that makes it easily distinguishable which stage of development this is.

Now since you created a branch you need to create a central repository. The central repository can be on a server or any other location you choose.

1. Create a new folder and call it for exemple "Repository".
2. Open the folder right-click and select "GitExt Create new repository...".
3. Select "Central repository"

You have created your central repository.

Now you need to link the central repository to your exisiting development stage and the corresponding branch.

1. Go into "Git Browse" GUI select your branch and
2. Click on Repository –> Remote repositories
3. Choose a name of your liking
4. Enter the URL of the created central repository.

After linking the branch with the central repository you need to "push" the contets of the branch to the central repository. This is done by selecting the blue arrow in the Git browse that point up.

18.2.4 Creating the different development stages

The following steps should be repeated for every stage of development you want to set up.

1. Copy the folder of your already established devlopment stage.

!! It is very important that you do this **outside** of your workflow root directory. If you do not do this *kenaflow* will run into errors since it recognizes the "new" workflow and runs it. Due to the fact that both (the original and the copy) have the same ID *kenaflow* runs into an problem and throws an error for one of these workflows. To prevent this copy the folder outside ot the root directory and change the ID and all the other necessary entries.

The advantage of copying the folder is that the underlying structure (the connection of the personal and central repository) is maintained.

1. Create a new branch for the newly created development stage in the Git browse.
2. Push the contents of the new branch.

18.3 Workflow Error When Using Invoke-WebRequest

When you use the PowerShell cmdlet `Invoke-WebRequest` in a workflow script you could receive this error:

```
The response content cannot be parsed because
the Internet Explorer engine is not available,
or Internet Explorer's first-launch configuration
is not complete. Specify the UseBasicParsing parameter
and try again.
```

This indicates that "Internet Explorer" was not started for the **first time** by the executing user.

This user could be the *kenaflow* service account if the workflow runs with such an account, e.g. as scheduled task.

You have two options.

First try to use parameter `-UseBasicParsing` with `Invoke-WebRequest` to prevent the usage of Internet Explorer engine in the request.

If that is not sufficient please login on the *kenaflow* server once with the *kenaflow* service account, start Internet Explorer and finish the "first-launch configuration".

If it fails in debug you probably did not start Internet Explorer with your own user account once. Please do so and finish the "first-launch configuration".

18.4 Get/Set SharEPoint List Item Moderation State

You can read the SharePoint list item moderation state using this command:

```
$moderationState = $item["_ModerationStatus"]

switch($moderationState) {
  0 {
    #Approved
  }

  1 {
    #Denied
  }

  2 {
    #Pending
  }

  3 {
    #Draft
  }

  4 {
    #Scheduled
  }
}
```

You can set the moderation state from within a workflow using the cmdlet *`Set-KFFinalModerationState`*

CHAPTER

NINETEEN

OPEN SOURCE LICENSES

Included Open Source software and libraries.

kenaflow uses a number of Open Source libraries and software.

Office 365 Dev PnP Core component

Project Homepage: https://github.com/SharePoint/PnP-Sites-Core

License: MIT > https://github.com/SharePoint/PnP-Sites-Core/blob/master/LICENSE

SharePoint PnP Powershell

Project Homepage: https://github.com/SharePoint/PnP-PowerShell

License: MIT > https://github.com/SharePoint/PnP-PowerShell/blob/master/LICENSE

Visual Studio Code

Project Homepage: https://github.com/Microsoft/PnP-PowerShellVSCode

License: MIT > https://github.com/Microsoft/vscode/blob/master/LICENSE.txt

---

Bouncy Castle C#

Project Homepage: http://www.bouncycastle.org/csharp

License: MIT "our license is an adaptation of the MIT X11 License http://opensource.org/licenses/mit-license.php and should be read as such." http://www.bouncycastle.org/csharp/licence.html

---

CefSharp

Project Homepage: https://github.com/cefsharp/CefSharp

License: CefSharp > https://github.com/cefsharp/CefSharp

CommandLineParser

Project Homepage: https://github.com/gsscoder/commandline

License: MIT > https://github.com/gsscoder/commandline/blob/master/License.md

---

DataTables

Project Homepage: https://datatables.net/

License: MIT > https://datatables.net/license/mit

HtmlAgilityPack

Project Homepage: http://html-agility-pack.net and https://github.com/zzzprojects/html-agility-pack

License: MIT > https://github.com/zzzprojects/html-agility-pack/blob/master/LICENSE

---

jQuery

Project Homepage: https://jquery.com/

License: MIT > https://jquery.org/license/

JSZip

Project Homepage: https://stuk.github.io/jszip/

License: MIT > https://github.com/Stuk/jszip/blob/master/LICENSE.markdown

---

MessagePack

Project Homepage: https://github.com/neuecc/MessagePack-CSharp

License: MIT > https://github.com/neuecc/MessagePack-CSharp/blob/master/LICENSE

Microsoft.NET

Project Homepage: https://www.microsoft.com/net/

License: MICROSOFT SOFTWARE LICENSE TERMS and MICROSOFT .NET LIBRARY > https://www.microsoft.com/net/dotnet_library_license.htm

---

Microsoft PowerShell 5 Reference Assemblies

Project Homepage: https://www.nuget.org/packages/Microsoft.PowerShell.5.ReferenceAssemblies/

License: MIT > https://github.com/PowerShell/PowerShell-Tests/blob/master/LICENSE

Pdfmake

Project Homepage: http://pdfmake.org and https://github.com/bpampuch/pdfmake

License: MIT > https://github.com/bpampuch/pdfmake/blob/master/LICENSE

StackExchange.Redis

Project Homepage: https://github.com/StackExchange/StackExchange.Redis

License: MIT > https://github.com/StackExchange/StackExchange.Redis/blob/master/LICENSE

SQLite

Project Homepage: http://sqlite.org/

License: Public Domain > https://www.sqlite.org/copyright.html

---

System.Data.SQLite

Project Homepage: https://system.data.sqlite.org

License: Microsoft Public License (MsPL) > https://system.data.sqlite.org/index.html/doc/trunk/www/copyright.wiki

---

Cronos

Project Homepage: https://github.com/HangfireIO/Cronos

License: Microsoft Public License (MsPL) > https://github.com/HangfireIO/Cronos/blob/master/LICENSE

YamlDotNet

Project Homepage: https://github.com/aaubry/YamlDotNet

License: Microsoft Public License (MsPL) > https://github.com/aaubry/YamlDotNet/blob/master/LICENSE.txt

---

PnP Core

Project Homepage: https://github.com/pnp/pnpcore

License: https://github.com/pnp/pnpcore/blob/master/LICENSE

PnP Framework

Project Homepage: https://github.com/pnp/pnpframework

License: https://github.com/pnp/pnpframework/blob/master/LICENSE

PnP PowerShell

Project Homepage: https://github.com/pnp/powershell

License: https://github.com/pnp/powershell/blob/master/LICENSE

VSCode PowerShell Add-In

Project Homepage: https://marketplace.visualstudio.com/items?itemName=ms-vscode.PowerShell

License: https://github.com/PowerShell/vscode-powershell/blob/master/LICENSE.txt

CHAPTER

TWENTY

PRIVACY POLICY

The *kenaro* GmbH (in short: "we") is the operator of this website, which serves the sale of the software "*kenaro*", and in addition, serves the users and partners with information about the software supplied. We respect the protection of privacy and comply with the legal data protection regulations. In the following we would like to inform you about how we handle your personal data.

1. Who is the responsible body?

The responsible body in sense of the legal data protection regulations is

kenaro GmbHPaul-Gruner-Str. 604107 LeipzigGermanyinbox =at= kenaro.comTelephone: +49 / 341 / 600 15 0990FAX: +49 / 341 / 600 15 0999

Managing director: Ingo KarsteinRegister court: Amtsgericht Leipzig, HRB 31878

2. When and for what purpose does *kenaro* GmbH collect personal data?

** *kenaro* GmbH ** collects, stores or processes data exclusively for its own business purposes.

Our websites are always available to all users without personal information being collected.

Personal data is only requested, processed and used to that extent, which is necessary for offering you our software and services.

In particular, we collect data about the sale of our software products to the purchasers (end customer), intermediaries or distribution partners.

For the licensing and activation process of our software products we store contact information for each license sold in particular the name and the e-mail address of the end customer. If the end customer is a company, then the end customer can create an anonymous e-mail inbox using the company's name for the contact, instead of an employee's name and e-mail address. When individuals order software licenses, their name and e-mail address are stored.

We delete license information and related personal information data 1 year after expiry of the license if it has not been renewed.

We use the contact information you provide when you purchase a license to send emails to using this contact information: i.e. When the license is activated, when the license is deactivated, before the license expires, when the license expires.

Private persons are able to subscribe to the "newsletter" on our websites. When registering, we store personal data (name and e-mail address and, if need be, options for ordering the newsletter) in order to be able to deliver the newsletter(s) ordered.

3. Collecting general information

When you access our websites, information of a general nature is automatically collected. This information includes: Name and version of the web browser used, the operating system used, the domain name of your Internet service provider and the like. This is only information that does not allow any conclusions to be drawn about you as a person. This data is also requested when accessing other websites on the Internet. It is therefore not a special function of our website. The above information is statistically evaluated by us.

4. Dealing with your personal information

We protect your personal data against unauthorized access and from abuse. We act in accordance with the data protection law Provisions, as far as possible. We will be bringing up all necessary, technical and organizational security efforts to address this guarantee.

To protect the security of your data during transmission, we use encryption methods (e.g. SSL) over HTTPS. Our servers are secured by a firewall and virus protection.

5. Use of Cookies

So-called "cookies" are sometimes used on our websites. These are very small text files which are stored by your browser on the device you are using and which, among other things, make it possible to make visiting a website more comfortable or safer. Cookies can also be used to better match the offer on a website to the interests of the visitors or to improve it in general on the basis of statistical evaluations.

You can decide for yourself if the browser you are using allows the use of cookies or not. Please note that the functionality of websites may be restricted or even suspended when the use of cookies are not allowed.

6. Use of Google Analytics

Our websites use Google Analytics, a web analytics service of the Google Inc. ("Google"). Google Analytics uses so-called "cookies", which are text files, that will be stored on your computer and can analyze your use of the website by us. The cookie-generated information about your use of this website will be usually transmitted to a Google server in the US and also stored there. In case of activation of IP anonymization on this website, however, your IP address will be stored shortened by Google within member states of the European Union or in other contracting states of the agreement on the European Economic Area. Only in exceptional cases the full IP address will be transferred to a Google server in the USA and shortened there. On behalf of the operator of this website Google will use this information to improve your use of this website, to evaluate reports on website activity and to provide further information about the use of the website and the internet use of related services and pass these informations to the website operator. The data collected within the framework of Google Analytics by the transmitted IP address of your browser will not be combined with other data by Google. You may refuse the use of cookies by selecting the appropriate settings on your browser. However, please note that if you do this, you may not be able to use all the features of this website to the fullest possible extent. You can also disable the capture and processing of data through the cookie (including your ip address) by downloading and installing the following browser-plugin (http://tools.google.com/dlpage/gaoptout?hl=de)

As an alternative to the browser add-on described above or within browsers on mobile devices, you can also stop tracking through Google Analytics by clicking [this link](javascript:(function(){gaOptout(); alert("Google Analytics Opt-out-Cookie erfolgreich gesetzt.");})();). An opt-out cookie will be installed on your device. This will prevent the collection of data by Google Analytics for this website browser in the future, as long as the cookie in your browser remains installed.

Further information can be found under the following sites http://tools.google.com/dlpage/gaoptout?hl=de and http://www.google.com/intl/de/analytics/privacyoverview.html (general information about Google Analytics and data protection).

We would like to point out that at all of *kenaro* GmbH operated websites Google Analytics has been enhanced with the follwing code "Gat.anonymizeIp();" to include an anonymous capture IP addresses (so-called IP masking).

7. Social Plug-Ins from Facebook, Twitter, Google+, LinkedIn and Xing

On our web pages we offer the possibility to use the following "Social Plugins" of the following companies:

- Facebook Inc., 1601 S. California Ave, Palo Alto, CA 94304, USA;
- „Tweet"-Button from Twitter Inc., 795 Folsom St., Suite 600, San Francisco, CA 94107, USA;
- "+1"-Button from Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA;
- „Recommended-Button" from LinkedIn Corporation, 2029 Stierlin Court, Mountain View, CA 94043, USA;
- „Share-Button" from XING AG, Gänsemarkt 43, 20354 Hamburg, Deutschland.

To protect your data, we rely on the solution of "Shariff". This will display the plugins on the website only as a graphic, which links to the the corresponding website of the plugin provider. By clicking of the graphic you will thus be forwareded to the respective services of the providers. Only then will your data be transferred to

the respective services. If you do not click on the graphic, there will be no exchange between you and the social networks mentioned above.

You can find Information about the terms of collecting and using of your data in the social networks, on their respected websites. Find more information about the Shariff solution here: here.

8. Comments feature

We offer a comments feature on our website if necessary. Users of our web pages can post messages and comments there.

So that a user can leave such a comment on one of our websites, a registration is necessary. This happens through registering with an Email address, username and password. In some circumstances also through registration through an third-party authentification process, e.g. Google Inc., Facebook Inc. or Microsoft Corp.

With registration through a username and password on our website, we will save these personal informations, combined with the email-address and, if possible, with the real name of the user. We do this to protect ourselves against misappropriated content on our website. This way we can trace back the users who have left illegal content on our websites.

When registering by means of an authentication procedure, the respective provider of the procedure is responsible for data protection. In this case, we store only the data of the user reported by the provider of the authentication procedure, in particular the name and e-mail address of the user, and additional technical information, e.g. registration token.

9. Contacting the *kenaro* GmbH

If a website user contacts us by e-mail or contact form, the information he has provided will be stored for the purpose of processing the request. We will contact the user if this the request makes it necessary.

10. How long does the data stay stored

We adhere to the principles of data avoidance and data economy. Therefore, we store personal data only as long as necessary to achieve the purposes stated here or as required by the legislator by specifying mandatory storage periods. After discontinuation of the respective purpose or expiration of these deadlines, the corresponding data are routinely blocked and deleted in accordance with the statutory provisions.

Regarding our software products, we store licensing information for up to one year after expiration of a license, if it has not been renewed.

When users use and subscribe to our comments function, we store personal information about the user as long as the user is registered on our platforms and their comments are visible.

11. Your rights concerning information, correction, suspension, deletion and objection

You have the right of being informed about your personal data stored by *kenaro* GmbH at any time. You have a right to correction, blocking or, apart from the prescribed data storage for business transactions, deletion of your personal data.

Therefore please contact inbox@kenaro.com.

For a data lock to be taken into account at all times, this data must be stored in a lock file for control purposes. You can also request the deletion of the data, as long as there is no legal archiving obligation. As far as such an obligation exists, we lock your data on request.

You can make changes or revoke your consent for the future by notifying us.

12. Change of our privacy policy

We reserve the right to amend this privacy policy to always comply with the current legal requirements or to implement changes to our services in the privacy policy, e.g. when introducing new services or products.

Your new visit will then be subject to the new privacy policy.

13. Questions to the data protection commissioner

If you have further question about the data protection at the *kenaro* GmbH, please write an email or contact us:

kenaro GmbHPaul-Gruner-Str. 604107 LeipzigGermanyinbox =at= kenaro.comTelephone: +49 / 341 / 600 15 0990FAX: +49 / 341 / 600 15 0999

CHAPTER

TWENTYONE

IMPRINT

21.1 Contact Information

Paul-Gruner-Str. 6, 04107 Leipzig, GermanyPhone: +49 (0) 341 600 150 99 0Fax: +49 (0) 341 600 150 99 9Email: admin@kenaflow.comCompany Registration: Amtsgericht Leipzig, HRB 31878Company Location: Leipzig

Managing Director: Ingo Karstein

21.2 Responsible for web site content

kenaro GmbH, Paul-Gruner-Str. 6, 04107 LeipzigResponsible for the content: Ingo Karstein

21.3 Disclaimer

21.3.1 Liability for Content

We make every effort to keep the information on our Web site current, but accept no liability whatsoever for the content provided.

Pursuant to §7 par. 1 of TMG (German Tele-Media Act), the law limits our responsibility as a service provider to our own content on these Web pages.

According to §§8 to 10 of TMG, we are not obligated to monitor third party information provided or stored on our Web site. However, we shall promptly remove any content upon becoming aware that it violates the law. Our liability in such an instance shall commence at the time we become aware of the respective violation.

21.3.2 Liability for Links

Our site contains links to third-party Web sites. We have no influence whatsoever on the information on these Web sites and accept no guaranty for its correctness. The content of such third-party sites is the responsibility of the respective owners/providers.

At the time third-party Web sites were linked to ours, we found NO GROUNDS WHATSOEVER of any likely contravention of the law. We shall promptly delete a link upon becoming aware that it violates the law.

21.3.3 Copyrights

The content and works provided on these Web pages are governed by the copyright laws of Germany. Duplication, processing, distribution, or any form of commercialization of such material beyond the scope of the copyright law shall require the prior written consent of its respective author or creator.

CHAPTER

TWENTYTWO

GLOSSARY

22.1 *kenaflow* Instance

kenaflow can be installed on a computer as often as you like (except in the FREE edition).

Theoretically, different versions of *kenaflow* can be used without these different versions coming into conflict with each other.

Each instance has its own global configuration file and its own workflow root directory.

Each instance has a unique ID that is stored in the global configuration file.

Each instance is assigned a data directory. The default path is: `%ProgramData%\kenaflow\instance_<instance-id>`

22.2 Program Mode

`kenaflow.exe` is started multiple times with different program switches. Thats how the special job of each process is specified. These "jobs" we call "program mode". E.g. if *kenaflow* is started with `kenaflow.exe --run` we call the process "main mode". Please read this article: *The Details*.

22.3 Execution Lane

Workflows are executed in parallel. There is a configurable number of parallel workflows. If the maximum number of processes is reached the next workflow in the queue must wait. We call this "execution lanes". *The details are here.*

22.4 Execution Slot

Execution lanes can be grouped. Read this: *Execution Slots*.

22.5 Workflow Priority

One workflow can be "more business critical" than another. It's necessary that it runs more often. You can configure it's priority therefore.

22.6 TBE = Time Between Execution

It's the amount of time in seconds that must elapse between two **starts** of one workflow. If TBE is 0 it will be queues to "run again ASAP". If TBE is < 0 it will never run by time.

22.7 PowerShell ISE

"PowerShell Integrated Scripting Environment" A Software that is part of Windows OS, both the client operating system and the Windows Server. It may need to be activated as an optional feature in Windows.

For older versions of Windows there are updates via Windows Update or as download of the "Windows Management Framework".

kenaflow requires PowerShell 5.

22.8 Workflow Folder (Workflow Directory)

The workflow directory contains all the files required to execute a workflow. Normally the workflow configuration file `_wfconfig.ps1`. Possibly, other script files or other files that you create.

22.9 Workflow Root Folder (Workflow Root Directory)

The workflow root directory contains all workflow directories, but no workflows or workflow configuration file.

22.10 Workflow Configuration File

This is the file `_wfconfig.ps1`, which describes a workflow.

22.11 Global Configuration File

This file is unique for each *kenaflow* instance and is stored in the program directory of the *kenaflow* instance. This file is called `kenaflow.conf`

22.12 Data Folder (Data Directory)

The data directory of each *kenaflow* instance stores data and information relevant to the *kenaflow* instance.

Under the data directory are:

- Lock directory
- Skip directory
- Log directory
- Queue directory
- Mails directory

Each of these sub folders can be configured individually using a global configuration file.

For example, it makes sense to set the queue directory for different instances of *kenaflow* to an identical directory (using a global configuration file). Furthermore, in this scenario a separate *kenaflow* instance is configured solely for the acquisition of remote events. All other instances on the computer receive remote events via the shared queue directory.

22.13 Queue Folder (Queue Directory)

The remote events are stored in this directory for later processing.

22.14 Lock Folder (Lock Directory)

In this directory, *kenaflow* locks workflow processing by other instances using 0 byte files.

The processing of list elements by other instances is also prevented, so that *kenaflow* workflows do not overwrite each other's values in list elements.

22.15 Mails Folder (Mails Directory)

Here e-mails are saved as text files if the corresponding function is active, i.e. if *kenaflow* is instructed by the configuration globally or on workflow level (`_wfconfig.ps1`) not to send e-mails via SMTP, but to save them.

22.16 Log Folder (Log Directory)

Log files are stored here.

22.17 Skip Folder (Skip Directory)

Currently not used.

22.18 Remote Event

If elements are changed or created in SharePoint lists, *kenaflow* can be informed about it. SharePoint provides an interface for applications to register a web endpoint for corresponding information messages.

After registration, SharePoint contacts this web endpoint and reports which item was created or changed in which list.

This mechanism works with all three supported SharePoint versions: 2013, 2016, 2019 and Online.

It is also possible to send custom data to *kenaflow* workflows.

Please read article *Remote Event* for details.

22.19 Cmdlet

It is pronounced as: "Command-let". These are the PowerShell commands, a combination of a verb and a noun like `Get-Command`.

22.20 PowerShell

Is a script language from Microsoft, which has been available since about 2005 and is becoming increasingly important for the administration of Windows systems and Microsoft products.

PowerShell is based on the Microsoft.NET framework and can process all .NET libraries in addition to its own libraries. You can create your own PowerShell extensions. We have done this excessively and are thrilled about the possibility!

22.21 netsh.exe

. . . is a command line program from Windows for the control of some network functions on the operating system.

22.22 kenaflow.exe

. . . is the executable file of *kenaflow*.

22.23 kenaflow.runtime.dll

. . . is a .NET assembly required to load *kenaflow* outside of the *kenaflow* runtime. The assembly is also used to load the SharePoint PnP PowerShell module, whose installation on a *kenaflow* computer is not supported due to impending interference with *kenaflow*.

CHAPTER

TWENTYTHREE

FINAL NOTES

23.1 About this document

This document as well as the software "*kenaflow*" are copyrighted and property of **kenaro GmbH**.

kenaflow is registered trademark.

kenaro is registered trademark.

kenaflow is a product of **kenaro GmbH**

kenaro GmbH GmbH Paul-Gruner-Str. 6 | 04107 Leipzig | Germany

23.2 Disclaimer

We do not accept any liability for damages resulting from the use of the software.

kenaflow can be used to develop workflows. Workflows are computer programs / software. If these programs are incorrectly developed by a programmer, they may be able to lead to loss of data or even damage IT systems.

23.3 Brands

"Microsoft", "SharePoint", "PowerShell" and others are trademarks of Microsoft Corporation.

CHAPTER

TWENTYFOUR

KENAFLOW CHANGE LOG

All notable changes to *kenaflow* will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

24.1 [4.0.23] - 2021-11-07

24.1.1 Added

- Hash based flood protection for error mails.
- Basic implementation for Serilog as alternative logging. This is disabled by default.

24.2 [4.0.22] - 2021-11-04

24.2.1 Added

- Export Barcodes as SVG with `New-KFBarcode`
- Inject pre-defined variables in workflow scripts
- Add global configuration setting: `instanceName`

24.3 [4.0.21] - 2021-10-21

24.3.1 Changed

- Update PnP-Core to 1.4
- Update PnP-Framework to 1.7
- Update PnP-Powershell to 1.8
- Update VSCode to 1.16.0

24.4 [4.0.20] - 2021-10-08

24.4.1 Fixed

- Fixed bug in global config reader/parser

24.5 [4.0.19] - 2021-06-09

24.5.1 Fixed

- Fixed bug that occured when debugging a “ustructured data” remote event in a SharePoint workflow.

24.6 [4.0.18] - 2021-05-24

24.6.1 Fixed

- Fixed bug in CRON based execution

24.7 [4.0.17] - 2021-04-12

24.7.1 Fixed

- Bug in cmdlet Start-KFSharePointModule fixed
- In some cases the registration of HTTP endpoints fail without any plausible reason. We have implemented a workaround for this anyway.

24.8 [4.0.16] - 2021-03-05

24.8.1 Added

- Error reporting web hook (global configuriation / Monitoring)
- Free disk space monitoring (global configuriation / Monitoring)

24.8.2 Changed

- In cmdlet `Send-KFMail` duplicate mail addresses in TO and CC are filtered. But TO and CC are handled separately!

24.9 [4.0.15] - 2021-03-03

24.9.1 Fixed

- Bug fix in PDF creation

24.10 [4.0.14] - 2021-02-27

24.10.1 Fixed

- Fixed another memory leak bug.

24.11 [4.0.13] - 2021-02-01

24.11.1 Fixed

- Memory bug fixed

24.12 [4.0.12] - 2021-01-11

24.12.1 Added

- Queueing PowerShell sessions in Remote Event Runner. Please read this article: Global Configuration, Section "Runtime"
- New cmdlets `Get-KFSPDatetimeLocal` and `Set-KFSPDatetimeLocal`

24.13 [4.0.11] - 2020-12-24

24.13.1 Added

- New Cmdlets `Update-KFExcelFile` and `Import-KFExcelFile`
- New cmdlets `Set-KFObject`, `Get-KFObject` and `Remove-KFObject`
- New workflow option `PreProcessingScript` and `PostProcessingScript`

24.14 [4.0.10] - 2020-12-18

24.14.1 Added

- New Cmdlet `New-KFExcelFile`

24.15 [4.0.9] - 2020-11-10

24.15.1 Added

- New option for "additional config file" in _wfconfig.ps1 for serialization control

24.16 [4.0.8] - 2020-11-09

24.16.1 Fixed

- Bugfix in licensing component. License root cert check failed with this previsous installer because the root cert was not installed correctly.

24.16.2 Changed

- VSCode: 1.5.1.0
- PnP: 3.26.2010.0 October 2020

24.17 [4.0.7] - 2020-09-28

24.17.1 Changed

- Now a runtime culture can be set in global configuration ("culture"). This culture is by default "en-US" (LCID: 1033). Is is used in debug mode and automatic mode. Before the users culture could be different to the service accounts culture.

24.17.2 Added

- Pipeline output can know be written to the log using global configuration setting "logOutput". Default: false

24.18 [4.0.6] - 2020-09-14

24.18.1 Fixed

- Fixed bug in storage: wrong passphrase used if additional passphrase is specified.

24.18.2 Added

- New build branch for diagnostics introduced

24.19 [4.0.5] - 2020-07-29

24.19.1 Changed

- PnP PowerShell update: now version 3.22.2006.2 is included (embedded)
- Embedded VSCode updated to version 1.47.2
- New command line parser for performance improvements
- New native data serializers for performance improvementsj

24.19.2 Fixed

- Meaningful error message if there are duplicate entries in the State List for State Machine Workflows

24.20 [4.0.4] - 2020-01-31

24.20.1 Changed

- VSCode updated to version 1.43.2
- PNP updated to 3.19.2003 (March 2020)

24.21 [4.0.0] - 2020-01-31

NEW MAJOR VERSION.

24.21.1 Changed

- As a new prerequisite *kenaflow* 4.0 now requires .NET Framework 4.8.
- *kenaflow* now supports "Agents" for remote PowerShell execution in a highly secure fassion from within workflows

24.22 Added

- *kenaflow Agent* - See doc.

24.23 [3.0.17] - 2019-12-12

24.23.1 Added

- Now there is a new remote event receiver checker in the main mode. This is enabled by default but can be disabled in the global configuration.

24.24 [3.0.16] - 2019-12-11

24.24.1 Fixed

- There was a (newly introduced) bug in the global remote event receiver program mode.

24.25 [3.0.15] - 2019-11-26

24.25.1 Fixed

- The was a bug when debugging a state machine workflow. When trying to **debug** a script that was not correctly registered as a state script this causes an execution of all correctly registered state scripts.

24.26 [3.0.14] - 2019-09-17

24.26.1 Changed

- Before this version workflows and (SharePoint state machine workflow) states have been enabled/disabled by using config property `Ignore`. This was replaced by property `Enabled`. It is more handy. `Ignore` will work also but you get a warning in the log.
- program switch `--registersslcertforrer` was renamed to `--confighttp`. The old one will work too.
- with `--confighttp` it's now possible to bind the SSL certificate to an IP/port *or* hostname/port endpoint.
- In global configuration you can control the RER endpoint much better. you can specify a hostname with `rerHost` or an IP address with `rerHostIp`. If nothing is specified the http endpoints are bound to all IP addresses for the given port. If a host is specified with `rerHost` the http endpoint is bound to this host name with `server name indication` feature of Windows >= 8 / Windows Server >= 2012. **SNI** is not available on Windows < 8 / Windows Server < 2012. Currently the OS version is not checked by the engine.

24.27 [3.0.13] - 2019-09-17

24.27.1 Added

This version is kind of a major release...

- We implemented "Plugins": *Plugins*
- We implemented "Storage": *Storage*

24.27.2 Changed

- Massive performance improvements. We refactored some thread handling and synchronization code.
- We included Cronos library (https://github.com/HangfireIO/Cronos/blob/master/LICENSE) for CRON handling and remove our own CRON parser.
- Package update for several embedded Open Source libraries.
- Embedded Visual Code update to 1.38

24.28 [3.0.12] - 2019-09-08

24.28.1 Changed

- Now there can be multiple "custom libraries" in `_wfconfig.ps1`.

24.29 [3.0.11] - 2019-08-20

24.29.1 Fixed

- Bug fix in permission cmdlets: error occurred while resolving user objects to principal objects that are invalid in SharePoint

24.30 [3.0.10] - 2019-07-29

24.30.1 Fixed

- Bug fix in new ClientConnection wrapper (introducted in v3.0.8).

24.31 [3.0.9] - 2019-07-26

24.31.1 Fixed

- Bug fix in email workflow configuration: if an email workflow is not connected to PowerShell it will throw an error during `_wfconfig.ps1` analyzation. This bug was introduced with an improvement in v3.0.6

24.31.2 Changed

- Item recheck for SharePoint list and state machine workflows has been changed. After the configured amount of time the ids of the list items matching the workflow query are reloaded, instead of checking each item before processing it (old machanism).

24.32 [3.0.8] - 2019-07-25

24.32.1 Fixed

- We noticed a strange behaviour of `Microsoft.SharePoint.Client.dll` - the official client library for .NET for SharePoint. In case of timeouts it's not possible to execute the query again because after the exception the query is empty. Calling `ClientContext.Execute()` a second time after an exception is not working. So created a wrapper class arount ClientContext to handle this "bug" - as we call it.

24.32.2 Changed

- We updated the embedded Visual Studio Code to version 1.36.1. (https://code.visualstudio.com/updates/v1_36)

24.33 [3.0.7] - 2019-07-24

24.33.1 Fixed

- During email send it can happen that an exception of type "PropertyOrFieldNotInitializedException" is thrown when evaluating accounts. This has been fixed. But sometimes it can occure if an SharePoint account is damaged.

24.33.2 Changed

- If a workflow fails to connect to SharePoint multiple times the workflow will stop know.
- NuGet package updates

24.33.3 Added

- program switch `--ulsviewer` to start the ULSViewer with *kenaflow* live logs from the correct log file location.### Added

24.34 [3.0.6] - 2019-07-18

24.34.1 Changed

- for performance improvements we have changed the way we deliver necessary resources. they are embedded in kenaflow.data.dll. Before we compressed them but thats not so fast and we decided to store resources uncompressed. After some testing it feels good!

24.35 [3.0.5] - 2019-07-17

24.35.1 Changed

- In a SharePoint workflow of type *list* or *state machine* with the operation mode "last modified" the *kenaflow* field is used to persist some information the processing of the current item. In case of an error in the workflow script it could happen that this item is not processed again because the engine saves the processing state in the *kenaflow* field and skips the item next time. Beginning with *kenaflow* 3.0.5 we do not update the *kenaflow* field value in this cases:
  - An script error occured
  - An error was not handled by an error handling script or was re-thrown by the error handling script
  - The item processing was stopped during debug in a PowerShell debugger

24.35.2 Added

- For performance improvements *kenaflow* now saves a serialized version of the global configuration file - kenaflow.conf. This file is named `kenaflow.ser.conf`. It getting updated by the engine if necessary.

24.36 [3.0.4] - 2019-07-08

24.36.1 Added

- New feature for `_wfconfig.ps1`: With setting `RunCondition` it is possible to stop a workflow in an early state of processing using the config list. (General Workflow Settings)

24.36.2 Changed

- for SharePoint *state machine* workflow there is no longer a need to have a script file for ignored states
- for SharePoint *state machine* workflows an error is thrown if a (not ignored) state in the `_wfconfig.ps1` file has no entry in the SharePoint state list.
- for SharePoint *state machine* now there is a check for invalid characters in the state key, because they state keys are used for file names. If such a character is found - e.g. `\` the `_wfconfig.ps1` is invalid and the workflow will not execute.

24.36.3 Fixed

- Bug fix in error handling during debug related to `Stop-KFScript`.

24.37 [3.0.3] - 2019-06-26

24.37.1 Added

- New feature for `Send-KFMail`: Parameter `-RedirectMailMessage` was added to allow you to redirect processed mails in an *email workflow*. You can overwrite the mail properties by specifying the cmdlet parameters, e.g. -To -Subject -Body -IsHtmlMail -From -CC. If they are not specified in the cmdlet call they are taken from the redirected email.

24.38 [3.0.2] - 2019-06-25

24.38.1 Changed

- Email data in Mail workflows now are System.Net.Mail.MailAddress with DisplayName and Address

24.38.2 Added

- Email data in mail workflows new contain "Recipients" and "Mail Headers".

24.38.3 Fixed

- Bug fix in error handling: Errors that occured in the debugger were not visible in the console output window

24.39 [3.0.1] - 2019-06-24

24.39.1 Changed

- In all workflows the config list default values from `_wfconfig.ps1` are loaded before the SharEPoint list - if specified - is loaded. The default values are overwritten by the values loaded from the config list.

24.40 [3.0.0] - 2019-06-03

24.40.1 Added

- Remote Event debugging
- Return data from unstructured remote events
- Return message from link remote event

24.40.2 Changed

- Remove some RER settings from `_wfconfig.ps1` because they cannot be configured at workflow level (RER Host, RER Use SSL, RER Port)
- Objects that are written to the PowerShell pipeline that reach *kenaflow* as last recipient are not written to the file log. This is for performance reasons.
- Removed "ClientContext" parameter from cmdlets `Add-KFPermission`, `Set-KFPermission`, `Remove-KFPermission` and `Update-KFPermission`. If an list item is specified using parameter "Item" it's client context is used.
- Refactored error handling

24.40.3 Fixed

- Unknown placeholder could lead to long-running workflow scripts.

24.40.4 [3.0.0] - 2018-05-01

Start of implementation of **kenaflow 3.0**