ACCRUENT

ACCRUENT MERIDIAN

ENTERPRISE VBSCRIPT API REFERENCE

2026

Accruent, LLC 11500 Alterra Parkway, Suite 110, Austin, TX 78758 www.accruent.com

Accruent Confidential and Proprietary, copyright 2026. All rights reserved.

This material contains confidential information that is proprietary to, and the property of, Accruent, LLC. Any unauthorized use, duplication, or disclosure of this material, in whole or in part, is prohibited.

No part of this publication may be reproduced, recorded, or stored in a retrieval system or transmitted in any form or by any means—whether electronic, mechanical, photographic, or otherwise—without the written permission of Accruent, LLC.

The information contained in this document is subject to change without notice. Accruent makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Accruent, or any of its subsidiaries, shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Contents

Meridian Enterprise Introduction **53**
What's In This Guide **54**
Who Should Read This Guide **55**
Technical Support **56**
Meridian Enterprise Script Editor **57**
Edit VBScript Code 57
User Interface 59
Effects Of Custom Scripting **62**
Debugging VBScript **63**
Debug in Visual Studio 63
Host Applications You Can Use to Debug 65
Enable VBScript Logging and Viewing 65
Download Log with PowerWeb 66
Configuration Expressions **67**
Meridian Object Model **68**
Attachment Object **69**
Attachment Object Properties 69
FilePath Property 69
Syntax 69
Name Property 69
Syntax 69
Remarks 69
Attachments Object **70**
Attachments Object Properties 70
Attachment Property 70
Syntax 70
Count Property 70
Syntax 70
Attachments Object Methods 70
Add Method 70
Syntax 70
Return Value 71
Remove Method 71
Syntax 71
RemoveAll Method 71

Syntax 71
Parameters 71
Batch Object **72**
Batch Object Properties 72
Argument Property 72
Syntax 72
Remarks 73
ApplyToAll Property 73
Syntax 73
Remarks 73
BatchIndex Property 74
Syntax 74
BatchSize Property 74
Syntax 74
ConfirmationTitle Property 74
Syntax 74
Example 74
CountFailed Property 75
Syntax 75
CountSucceeded Property 75
Syntax 75
IsFirstInBatch Property 75
Syntax 75
IsLastInBatch Property 75
Syntax 75
ProcessAsBatch Property 75
Syntax 76
Batch Object Methods 76
Abort Method 76
Syntax 76
Remarks 76
AddDocuments Method 76
Syntax 76
Remarks 77
AddDocumentToBatch Method 77
Syntax 77

Remarks 77
AskConfirmation Method 77
Syntax 77
Example 78
AskInput Method 78
Syntax 78
Example 78
Confirmation Method 78
Syntax 79
Return Value 79
Remarks 79
FailCurrent Method 79
Syntax 79
Remarks 80
Input Method 80
Syntax 80
Return Value 80
Example 80
PrintDetails Method 80
Syntax 81
Remarks 81
ShowInfo Method 81
Syntax 81
Example 82
BCPropStorage Object 83
BCPropStorage Object Properties 83
LayoutsNames Property 83
Syntax 83
Remarks 83
BCPropStorage Object Methods 83
GetColumnDefs Method 84
Syntax 84
Return Value 84
Remarks 85
Property Method 85
Syntax 85

Return Value 85
Remarks 85
Client Object 86
Client Object Properties 86
Argument Property 86
Syntax 86
Remarks 86
BuildNr Property 87
Syntax 87
Remarks 87
ClientID Property 87
Syntax 87
ClientProductCode Property 87
Syntax 88
Confirmation Property 88
Syntax 88
Remarks 88
CurrentScope Property 88
Syntax 88
CurrentView Property 88
Syntax 89
Remarks 89
ImportDetails Property 89
Syntax 89
ImportType Property 89
Syntax 89
Type Property 89
Syntax 89
Viewer Property 89
Syntax 89
Client Object Methods 90
GoTo Method 90
Syntax 90
Remarks 90
GoToView Method 90
Syntax 90

Remarks 91
Log Method 91
Syntax 91
Remarks 91
NewMailMessage Method 91
Syntax 91
Parameters 92
Return Value 92
Remarks 92
OpenInApplication Method 92
Syntax 92
Remarks 92
Refresh Method 92
Syntax 92
Remarks 93
Document Object 94
Document Object Properties 95
BriefCase Property 95
Syntax 95
CanEditProperty Property 95
Syntax 95
ConcurrentEngineeringRuleProperty Property 95
Syntax 96
Created Property 96
Syntax 96
CreatedBy Property 96
Syntax 96
CWFManagers Property 96
Syntax 96
CWFState Property 96
Syntax 96
CWFTodoPersons Property 97
Syntax 97
DocumentType Property 97
Syntax 97
FileName Property 97

Syntax 97
Remarks 97
FileType Property 97
Syntax 97
ForgeURL property 98
Syntax 98
GlobalID Property 98
Syntax 98
HasIncomingReferences Property 98
Syntax 98
HasOutgoingReferences Property 98
Syntax 98
HasRedlines Property 98
Syntax 99
HasRendition Property 99
HasRenditionRedlines Property 99
Syntax 99
HybridParts Property 99
Syntax 99
ID Property 99
Syntax 99
ImportType Property 99
Syntax 100
IsChangedByPortal Property 100
Syntax 100
Example 100
IsUnderRevision Property 100
Syntax 100
IsUniqueValue Property 101
Syntax 101
LayerTranslationTable Property 101
Syntax 101
LayoutsNames Property 101
Syntax 101
Returns 101
Remarks 101

LockingProjectCopy Property 102
Syntax 102
MasterDocument Property 102
Syntax 102
Modified Property 102
Syntax 102
ModifiedBy Property 102
Syntax 102
ParentFolder Property 103
Syntax 103
ParentProject Property 103
Syntax 103
Path Property 103
Syntax 103
ProjectCopy Property 103
Syntax 103
Property Property 103
Syntax 104
Rendition Property 104
Syntax 104
RenditionStatus Property 104
Syntax 104
Remarks 104
Revision Property 104
Syntax 104
ShareID Property 104
Syntax 105
Size Property 105
Syntax 105
StatusText Property 105
Syntax 105
UnderRevisionBy Property 105
Syntax 105
VersionID Property 105
Syntax 105
WorkFlowAction Property 105

Syntax 106
WorkFlowManager Property 106
Syntax 106
WorkflowState Property 106
Syntax 106
WorkFlowTodoPerson Property 106
Syntax 106
Document Object Methods 107
AddRendition Method 107
Syntax 107
Remarks 107
AddReorderingComment Method 107
Syntax 107
ApplyPropertyValues method 108
Syntax 108
Parameters 108
AttachWaitingProjectCopy Method 108
Syntax 108
CalculateFileType method 108
Syntax 108
Parameters 108
CallRemote Method 109
Syntax 109
Example 109
ChangeDocumentType Method 110
Syntax 110
Remarks 110
ChangeForgeURL Method 110
Syntax 110
ChangeManager Method 111
Syntax 111
Remarks 111
ChangeTodoPerson Method 111
Syntax 111
Remarks 111
ChangeWorkflowState Method 112

Syntax 112
Remarks 112
ClearRedlines Method 112
Syntax 112
Parameters 112
CreateWaitingProjectCopy Method 113
Syntax 113
ConfirmMerged Method 113
Syntax 113
Parameters 113
ConfirmSuperceded Method 113
Syntax 113
Parameters 114
ContentFromReport Method 114
Syntax 114
CopyProperties Method 114
Syntax 114
Remarks 115
CreateHybridPart Method 115
Syntax 115
Return Value 115
Remarks 115
Delete Method 116
Syntax 116
Parameters 116
Remarks 116
DeleteHotspots method 116
Syntax 116
DeleteHybridPart Method 116
Syntax 116
Remarks 117
DeleteRendition Method 117
Syntax 117
Parameters 117
Remarks 117
ExecTransition Method 117

Syntax 117
Remarks 118
ExportRendition Method 118
Syntax 118
ExtractTags Method 118
Syntax 119
Return Value 119
Remarks 119
GetDocumentRevisions 119
Syntax 119
Parameters 119
Example 119
GetExpectedTransitionResult Method 120
Syntax 120
Return Value 120
GetHybridPart Method 120
Syntax 121
Return Value 121
GetLog Method 121
Syntax 121
Parameters 121
Return Value 121
Remarks 121
Example 121
GetNavigationViewLevelPath Method 122
Syntax 122
Return Value 122
GetReferences Method 122
Syntax 122
Return Value 122
HasHotspots method 123
Syntax 123
HybridMainDocument Method 123
Syntax 123
Return Value 123
Remarks 123

ImportHybridPart Method 123
Syntax 123
Return Value 124
Remarks 124
LinkToMaster Method 124
Syntax 124
Remarks 124
LoadFromFile Method 125
Syntax 125
Return Value 125
Remarks 125
LoadFromTemplate Method 125
Syntax 125
Remarks 125
LockLWS Method 126
Syntax 126
Parameters 126
Return Value 126
LockMasterDocument Method 126
Syntax 126
Parameters 126
Log Method 126
Syntax 126
Migrate Method 127
Syntax 127
MoveTo Method 127
Syntax 127
Remarks 128
ReassignManagers Method 128
Syntax 128
Remarks 129
ReassignTodoPersons Method 129
Syntax 129
Remarks 129
ReleaseChange Method 129
Syntax 129

Parameters 129
Remarks 130
Reroute Method 130
Syntax 130
Remarks 130
ResetPropertyValues Method 130
Syntax 130
Parameters 130
RevokeChange Method 131
Syntax 131
Parameters 131
Remarks 131
RevokeWorkflow Method 131
Syntax 131
Parameters 131
Remarks 131
SaveToFile Method 131
Syntax 132
Return Value 132
Remarks 132
SendToPortal Method 132
Syntax 132
Remarks 133
SetModified Method 133
Syntax 133
Remarks 133
StartChange Method 133
Syntax 133
Parameters 133
Remarks 134
StartWorkflow Method 134
Syntax 134
Remarks 134
Subscribe Method 134
Syntax 134
Unsubscribe Method 135

Syntax 135
UnlockLWS Method 135
Syntax 135
Parameters 135
Return Value 135
UnLockMasterDocument Method 136
Syntax 136
Parameters 136
UpdateRendition Method 136
Syntax 136
Remarks 136
DocumentType Object 137
DocumentType Object Properties 137
ConfigKeywords Property 137
Syntax 137
Remarks 137
Example 137
DisplayName Property 137
Syntax 137
InternalName Property 137
Syntax 138
Sequence Property 138
Syntax 138
Remarks 138
ExportPackage Object 139
ExportPackage Object Properties 139
ContentIncluded Property 139
Syntax 139
Return Values 139
Remarks 139
Description Property 139
Syntax 139
Remarks 139
Destination Property 140
Syntax 140
Remarks 140

ID Property 140
Syntax 140
Name Property 140
Syntax 140
Remarks 140
PackageURL Property 141
Syntax 141
ProjectID Property 141
Syntax 141
Remarks 141
ProjectName Property 141
Syntax 141
Recipient Property 141
Syntax 141
Remarks 141
Status Property 142
Syntax 142
Return Values 142
ExportPackage Object Methods 142
Abort Method 142
Syntax 142
Remarks 142
AddDocument Method 142
Syntax 142
Remarks 143
Close Method 143
Syntax 143
Remarks 143
GetDocuments Method 143
Syntax 143
Return Value 143
Remarks 143
IsDocumentInPackage Method 143
Syntax 143
Return Value 144
IsDocumentOutdatedInPackage Method 144

Syntax ... 144
Return Value ... 144
RemoveDocument Method ... 144
Syntax ... 144
Remarks ... 145
Reopen Method ... 145
Syntax ... 145
Remarks ... 145
Send Method ... 145
Syntax ... 145
Remarks ... 145
SetProjectID Method ... 145
Syntax ... 145
Remarks ... 145
Update Method ... 145
Syntax ... 146
Remarks ... 146
ExportPackages Object ... 147
ExportPackages Object Methods ... 147
CreateNewPackage Method ... 147
Syntax ... 147
Return Value ... 147
FindDocumentPackages Method ... 147
Syntax ... 147
Return Value ... 148
FindPackage Method ... 148
Syntax ... 148
Return Value ... 148
Folder Object ... 149
Folder Object Properties ... 150
AutoDocumentType Property ... 150
Syntax ... 150
Remarks ... 150
Created Property ... 150
Syntax ... 150
CreatedBy Property ... 151

Syntax 151
HasDocuments Property 151
Syntax 151
ID Property 151
Syntax 151
IsProject Property 151
Syntax 151
IsUniqueValue Property 151
Syntax 152
LinkedProjectID Property 152
Syntax 152
Name Property 152
Syntax 152
Remarks 152
ParentFolder Property 152
Syntax 153
ParentProject Property 153
Syntax 153
Path Property 153
Syntax 153
ProjectTypeDisplayName Property 153
Syntax 153
ProjectTypeName Property 153
Syntax 153
Property Property 153
Syntax 154
Roles Property 154
Sequence Property 154
Syntax 154
ShareID Property 154
Syntax 154
TypeDisplayName Property 154
Syntax 155
TypeName Property 155
Syntax 155
WorkflowManagers Property 155

Syntax 155
WorkFlowState Property 155
Syntax 155
WorkflowStatus Property 155
Syntax 155
Folder Object Methods 156
ApplyPropertyValues Method 156
Syntax 156
Parameters 156
ChangeFolderType Method 156
Syntax 156
Remarks 156
ConvertToProject Method 157
Syntax 157
CopyProperties Method 157
Syntax 157
Remarks 157
Delete Method 158
Syntax 158
ExecTransition Method 158
Syntax 158
Remarks 158
FindDocuments Method 158
Syntax 158
GetExpectedTransitionResult Method 159
Syntax 159
Remarks 159
GetSubFolder Method 159
Syntax 159
Return Value 160
Remarks 160
GetSubFolderNames Method 160
Syntax 160
Return Value 160
GetUsersWithRole Method 161
Syntax 161

Return Value 161
Interlock Method 161
Syntax 161
LinkToPortal Method 162
Syntax 162
Parameters 162
MoveTo Method 162
Syntax 162
NewFolder Method 162
Syntax 162
Return Value 163
Remarks 163
ReassignWorkflowManagers Method 163
Syntax 163
Remarks 163
Reroute Method 164
Syntax 164
ResetPropertyValues Method 164
Syntax 164
Parameters 164
UnlinkFromPortal Method 164
Syntax 164
Parameters 164
ImportPackage Object 165
ImportPackage Object Properties 165
Description Property 165
Syntax 165
Remarks 165
ID Property 165
Syntax 165
Name Property 165
Syntax 165
Remarks 165
ImportProfile Property 166
Syntax 166
ImportPackage Object Methods 166

GetDocuments Method 166
Syntax 166
Return Value 166
Remarks 166
ImportProfile Object 167
ImportProfile Object Properties 167
Destination Property 167
Syntax 167
ID Property 167
Syntax 167
Name Property 167
Syntax 167
Remarks 167
MailMessage Object 169
MailMessage Object Properties 169
Attachments Property 169
Syntax 169
NoteText Property 169
Syntax 169
Originator Property 169
Syntax 169
Recipients Property 169
Syntax 169
Subject Property 170
Syntax 170
MailMessage Object Methods 170
Clean Method 170
Syntax 170
Parameters 170
Send Method 170
Syntax 170
MailRecipient Object 171
MailRecipient Object Properties 171
Address Property 171
Syntax 171
Remarks 171

Name Property ... 171
Syntax ... 171
Remarks ... 171
Type Property ... 172
Syntax ... 172
MailRecipients Object ... 173
MailRecipients Object Properties ... 173
Count Property ... 173
Syntax ... 173
Recipient Property ... 173
Syntax ... 173
MailRecipients Object Methods ... 173
Add Method ... 173
Syntax ... 173
Return Value ... 174
Remarks ... 174
Examples ... 174
Remove Method ... 174
Syntax ... 174
RemoveAll Method ... 175
Syntax ... 175
Parameters ... 175
MasterDocument Object ... 176
MasterDocument Object Methods ... 176
TransferToNext Method ... 176
Syntax ... 176
Example ... 176
MeridianQueue Object ... 177
MeridianQueue Object Methods ... 177
BeginBatch Method ... 177
Syntax ... 177
Remarks ... 177
Example ... 178
CommitBatch Method ... 179
Syntax ... 179
Remarks ... 179

RegisterDocument Method ... 179
Syntax ... 180
Return Value ... 181
Remarks ... 181
RevokeBatch Method ... 181
Syntax ... 181
Remarks ... 181
MeridianTask Object ... 182
MeridianTask Object Properties ... 182
DocumentID Property ... 182
Syntax ... 182
FeedbackProperty Property ... 182
Syntax ... 182
JobCode Property ... 182
Syntax ... 182
Remarks ... 183
PreventDuplicate Property ... 183
Syntax ... 183
PublishOptions Property ... 183
Syntax ... 183
RenderOptions Property ... 183
Syntax ... 183
RevisionID Property ... 183
Syntax ... 183
UserName Property ... 184
Syntax ... 184
NewMailMessage Object ... 185
NewMailMessage Object Properties ... 185
Attachments Property ... 185
Syntax ... 185
NoteText Property ... 185
Syntax ... 185
Originator Property ... 185
Syntax ... 185
Recipients Property ... 185
Syntax ... 186

Subject Property ... 186
Syntax ... 186
NewMailMessage Object Methods ... 186
Clean Method ... 186
Syntax ... 186
Parameters ... 186
Send Method ... 186
Syntax ... 186
ProjectCopy Object ... 187
ProjectCopy Object Properties ... 187
IsActive Property ... 187
Syntax ... 187
MasterLocked Property ... 187
Syntax ... 187
HasConcurrentPCs Property ... 187
Syntax ... 187
ProjectCopy Object Methods ... 188
UnlinkFromMaster Method ... 188
Syntax ... 188
RequireMerge Method ... 188
Syntax ... 188
Parameters ... 188
UndoMakeObsolete Method ... 188
Syntax ... 188
Parameters ... 188
Query Object ... 189
Query Object Properties ... 189
DBConnection Property ... 189
Syntax ... 189
DisplayName Property ... 189
Syntax ... 189
Name Property ... 189
Syntax ... 189
Remarks ... 189
Type Property ... 190
Syntax ... 190

Query Object Methods ... 190
GetValues Method ... 190
GetValuesEx Method ... 190
References Object ... **191**
References Object Properties ... 191
Count Property ... 191
Syntax ... 191
Exist Property ... 191
Syntax ... 191
Properties Property ... 191
Syntax ... 191
Return Value ... 191
Example ... 193
Target Property ... 193
Syntax ... 193
References Object Methods ... 193
Add Method ... 193
Syntax ... 193
Remarks ... 194
Delete Method ... 194
Syntax ... 194
Remarks ... 194
Refresh Method ... 194
Syntax ... 194
Parameters ... 194
Report Object ... **195**
Report Object Properties ... 195
BriefCase Property ... 195
Syntax ... 195
DisplayName Property ... 195
Syntax ... 195
DocumentCount Property ... 195
Syntax ... 195
FileName Property ... 195
Syntax ... 195
Remarks ... 196

Format Property 196
Syntax 196
QuoteChar Property 196
Syntax 196
SepChar Property 196
Syntax 196
VaultName Property 196
Syntax 196
Roles Object 197
Roles Object Properties 197
Assignments Property 197
Syntax 197
Remarks 197
InheritedFromParent Property 197
Syntax 197
Remarks 197
Roles Object Methods 198
AddAssignment Method 198
Syntax 198
Remarks 198
Clear Method 198
Syntax 198
Parameters 198
Remarks 198
DeleteAssignment Method 199
Syntax 199
Remarks 199
Scope Object 200
Scope Object Properties 200
DisplayName Property 200
Syntax 200
Remarks 200
Name Property 200
Syntax 200
Remarks 200
DocumentTypes Property 201

Syntax 201
Remarks 201
RootFolder Property 201
Syntax 201
Remarks 201
Sequence Object 202
Sequence Object Properties 202
Next Property 202
Syntax 202
Value Property 202
Syntax 202
Sequence Object Methods 202
SetTo Method 202
Syntax 202
Return Value 203
StaticCollection Object 204
StaticCollection Object Methods 204
Add Method 204
Syntax 204
Remove Method 204
Syntax 204
RemoveAll Method 205
Syntax 205
Parameters 205
Table Object 206
Table Object Properties 206
DBConnection Property 206
Syntax 206
ColumnsInfo Property 206
Syntax 206
DisplayName Property 206
Syntax 206
Name Property 206
Syntax 206
Remarks 207
Type Property 207

Syntax 207
Table Object Methods 207
AddValues Method 207
Syntax 207
Clean Method 208
Syntax 208
Parameters 208
DeleteValues Method 208
Syntax 208
DeleteValuesEx Method 208
Syntax 208
GetValues Method 209
Syntax 209
Return Value 209
Remarks 209
GetValuesEx Method 211
Syntax 211
Return Value 211
Remarks 212
UpdateValues Method 213
Syntax 213
Remarks 213
UpdateValuesEx Method 213
Syntax 213
Remarks 214
Task Object 215
Task Object Methods 215
Reset Method 215
Syntax 215
Parameters 215
Remarks 215
Set Method 215
Syntax 215
Remarks 216
Submit Method 216
Syntax 216

Remarks 216
User object 217
User Object Properties 217
AllEmails Property 217
Syntax 217
Description Property 217
Syntax 217
EmailAddress Property 217
Syntax 217
FullName Property 218
Syntax 218
HasPrivilege Property 218
Syntax 218
HasRole Property 218
Syntax 218
Remarks 218
Initials Property 219
Syntax 219
Manager Property 219
Syntax 219
Name Property 219
Syntax 219
Remarks 219
OrganizationalUnit Property 220
Syntax 220
Profile Property 220
Syntax 220
Remarks 220
Title Property 220
Syntax 220
User Object Methods 221
Vault Object 222
Vault Object Properties 222
Argument Property 222
Syntax 222
Remarks 222

CurrentWorkArea Property 223
Syntax 223
HasFeature Property 223
Syntax 223
Moment Property 223
Syntax 223
Name Property 223
Syntax 223
Remarks 223
Option Property 224
Syntax 224
Path Property 224
Syntax 224
Query Property 224
Syntax 224
RootFolder Property 225
Syntax 225
RootURL Property 225
Syntax 225
Scope Property 225
Syntax 225
Sequence Property 225
Syntax 225
ServerName Property 226
Syntax 226
ServerProductCode Property 226
Syntax 226
ServerTimeGMT Property 226
Syntax 226
StaticCollection Property 226
Syntax 226
Remarks 227
Table Property 227
Syntax 227
Task Property 227
Syntax 227

User(Name) Property 227
Syntax 227
WebAccessLocationID Property 227
Syntax 227
WorkIsolationMode Property 228
Syntax 228
Vault Object Methods 228
AuditEvent Method 228
Syntax 228
Remarks 228
CallRemote Method 229
Syntax 229
Example 229
ComposeURL Method 230
Syntax 230
Return Value 230
Remarks 230
CreateNewDocument 230
Syntax 231
Return Value 231
Example 231
DisplayName Method 231
Syntax 232
Return Value 232
ExecSQL Method 232
Syntax 232
FindDocuments Method 233
Syntax 233
Return Value 233
Remarks 233
Example 235
GetDistinctValues Method 236
Syntax 236
Return Value 236
GetDocument Method 237
Syntax 237

Return Value 237
GetGroups Method 237
Syntax 237
Return Value 237
GetPropertyNames Method 238
Syntax 238
Return Value 238
GetUsers Method 238
Syntax 238
Return Value 238
RootFolder Method 239
Syntax 239
Return Value 239
RunShellCommand Method 239
Syntax 239
SendNotification Method 240
Syntax 240
Remarks 240
Viewer Object **241**
Viewer Object Properties 241
IsPreview Property 241
Syntax 241
Watermark Property 241
Syntax 241
Example 241
IsRendition Property 241
Syntax 241
WaitingList Object **242**
WaitingList Object Properties 242
Documents Property 242
Syntax 242
Count Property 242
Syntax 242
Document Property 242
Syntax 242
Property Property 242

Syntax 242
Priority Property 243
Syntax 243
WaitingList Object Methods 243
Move Method 243
Syntax 243
Remove Method 243
Syntax 243
Workflow Object 244
Workflow Object Properties 244
DisplayName Property 244
Syntax 244
Name Property 244
Syntax 244
Remarks 244
State Property 245
Syntax 245
Transition Property 245
Syntax 245
Remarks 245
WorkflowState Object 246
WorkflowState Object Properties 246
DisplayName Property 246
Syntax 246
Name Property 246
Syntax 246
Remarks 246
Type Property 247
Syntax 247
Workflow Property 247
Syntax 247
WorkflowTransition Object 248
WorkflowTransition Object Properties 248
CreateNewRevision Property 248
Syntax 248
DisplayName Property 248

Syntax 248
Name Property 248
Syntax 248
Remarks 248
SourceState Property 249
Syntax 249
TargetState Property 249
Syntax 249
Workflow property 249
Syntax 249
Meridian Functions 250
AIMS_Commands Function 251
Syntax 251
Parameters 251
Remarks 251
Example 251
AIMS_Properties Function 252
Syntax 252
Parameters 252
Remarks 252
Example 252
AIMS_UpdateChangeManagement Function 253
Syntax 253
Remarks 253
Example 253
AMCreateObject Function 254
Syntax 254
Remarks 254
AMMGetCustomColumnHeaders Function 255
Syntax 255
Parameters 255
Example 255
Remarks 255
AMMGetCustomColumnValues Function 256
Syntax 256
Example 259

Remarks ... 260
AMMGetReportCustomHeaderValues Function ... **261**
Syntax ... 261
Remarks ... 261
AMMGetReportTableRowValues Function ... **262**
Syntax ... 262
Remarks ... 266
AMMMainTagDocumentId Function ... **267**
Syntax ... 267
Parameters ... 267
Example ... 267
AMMPropertiesToBeRequested Function ... **269**
Syntax ... 269
Parameters ... 269
Example ... 269
Remarks ... 269
AMMTags4TagPageIsVisible Function ... **270**
Syntax ... 270
Parameters ... 270
Example ... 270
AMMTagsManageLinksIsAllowed Function ... **271**
Syntax ... 271
Parameters ... 271
Example ... 271
Remarks ... 271
AMMTagsPageIsVisible Function ... **272**
Syntax ... 272
Parameters ... 272
Example ... 272
AMMUseMainTag Function ... **273**
Syntax ... 273
Parameters ... 273
Example ... 273
AMMWhereUsedManageLinksIsAllowed Function ... **274**
Syntax ... 274
Parameters ... 274

Example ... 274
Remarks ... 274
AMMWhereUsedPageIsVisible Function ... 275
Syntax ... 275
Parameters ... 275
Example ... 275
DebugAssert Function ... 276
Syntax ... 276
Remarks ... 276
FileExtension Function ... 277
Syntax ... 277
Remarks ... 277
FileRoot Function ... 278
Syntax ... 278
FormatSequenceAlfa Function ... 279
Syntax ... 279
Remarks ... 279
FormatSequenceAlfaNum Function ... 280
Syntax ... 280
Remarks ... 280
FormatSequenceNum Function ... 281
Syntax ... 281
Remarks ... 281
GMTTime2Local Function ... 282
Syntax ... 282
ListFromColumn Function ... 283
Syntax ... 283
LocalTime2GMT Function ... 284
Syntax ... 284
PnIDLink_GetAssetCoordinates Function ... 285
Syntax ... 285
Return Value ... 285
Remarks ... 285
PnIDLink_GetSheetSize Function ... 286
Syntax ... 286
Remarks ... 286

PnIDLink_IsMainDrawing Function ... **287**
Syntax ... 287
Remarks ... 287
Quote Function ... **288**
Syntax ... 288
ValidateFolderName Function ... **289**
Syntax ... 289
Remarks ... 289
WinInputBox Function ... **290**
Syntax ... 290
Remarks ... 290
WinMsgBox Function ... **291**
Syntax ... 291
Remarks ... 291
Creating Custom Functions ... **292**
Meridian Event Procedures ... **293**
Names Of Events ... **294**
Batch Events ... **295**
Order Of Events ... **296**
Create and Edit Event Procedures ... **298**
Limiting Events Generated By VBScript ... **299**
Asset Management Events ... **300**
AIMS_AddComment Event ... 300
Syntax ... 300
Example ... 300
AIMS_Attach_Intialize Event ... 301
Syntax ... 301
Example ... 301
AIMS_Attach_Before Event ... 302
Syntax ... 302
Example ... 303
Remarks ... 304
AIMS_Attach_After Event ... 304
Syntax ... 304
Example ... 305
AIMS_DeleteComment Event ... 306

Syntax 306
Example 307
AIMS_CloseComment Event 307
Syntax 307
Example 307
AIMS_UpdateComment Event 308
Syntax 308
Example 308
ObjectsPage_IsVisible Event 309
Syntax 309
Remarks 309
Example 309
WhereUsedPage_IsVisible Event 309
Syntax 309
Remarks 309
Example 310
CAD Link Events 311
CAD Link Event Sequences 311
Synchronize properties from file event sequence 311
Synchronize properties to file event sequence 311
Synchronize references from file event sequence 312
AfterReadProperty Event 312
Syntax 312
Remarks 312
AfterReadMTBProperties Event 312
Syntax 312
Example 313
BeforeWriteProperty Event 313
Syntax 313
Remarks 314
BeforeWriteMTBProperties Event 314
Syntax 314
Example 314
OnUpdateReference Event 315
Syntax 315
*UpdateProperties Events 315

Syntax 315
*UpdateReferences Events 316
Syntax 316
Custom Command Events 317
Custom command event sequence 317
<CommandName>_State Event 317
Syntax 317
Remarks 318
<CommandName>_Initialize Event 318
Syntax 318
<CommandName>_BeforeWizard Event 318
Syntax 318
<CommandName>_AfterWizard Event 318
Syntax 318
<CommandName>_Execute Event 319
Syntax 319
Remarks 319
<CommandName>_PreExecute Event 319
Syntax 319
Remarks 320
<CommandName>_PreInitialize Event 320
Syntax 320
Remarks 320
<CommandName>_Terminate Event 320
Syntax 320
Custom Page Events 321
Custom Page Event Sequence 321
CustomPages_IsVisible Event 321
Syntax 321
CustomPages_CanEdit Event 321
Syntax 321
CustomPages_CanApply Event 322
Syntax 322
Document Copy/Move Events 323
Document Copy/Move Event Sequences 323
Copy, Paste, Derive, Replace event sequence 323

Move event sequence 324
AfterCopy Event 324
Syntax 324
AfterCopyWithReferences Event 325
Syntax 325
AfterDerive Event 325
Syntax 325
AfterMove Event 326
Syntax 326
AfterReplace Event 326
Syntax 326
*BeforeCopy Event 327
Syntax 327
Remarks 327
*BeforeCopyWithReferences Event 327
Syntax 327
Remarks 328
*InitializeCopy Event 328
Syntax 328
Remarks 329
InitializeDerive Event 329
Syntax 329
InitializeMove Event 329
Syntax 329
InitializeReplace Event 330
Syntax 330
PrepareCopy Event 330
Syntax 330
Example 330
SelectTarget Event 331
Syntax 331
Remarks 331
TerminateCopy Event 332
Syntax 332
TerminateDerive Event 332
Syntax 332

TerminateMove Event ... 333
Syntax ... 333
TerminateReplace Event ... 333
Syntax ... 333
Document Generic Events ... **334**
Document Generic Event Sequences ... 334
Change Document Type event sequence ... 334
Delete event sequence ... 334
Draft Print event sequence ... 334
Issue New Filename event sequence ... 335
New Document event sequence ... 335
Open in Application event sequence ... 336
Prepare for Offline Work event sequence ... 336
Redline event sequence ... 337
Refresh Thumbnail event sequence ... 337
Rename event sequence ... 337
Replace Content event sequence ... 337
AfterCreateReference Event ... 337
Syntax ... 338
AfterDeleteReference Event ... 338
Syntax ... 338
Remarks ... 338
AfterEditRedlines Event ... 339
Syntax ... 339
AfterReplaceContent Event ... 339
Syntax ... 339
BeforeCreateReference Event ... 339
Syntax ... 339
BeforeDeleteReference Event ... 340
Syntax ... 340
Remarks ... 340
BeforeEditRedlines Event ... 340
Syntax ... 341
BeforeReplaceContent Event ... 341
Syntax ... 341
Remarks ... 341

BeforeSelectDocType Event 342
Syntax 342
Remarks 342
*CalculateFilename Events 342
Syntax 342
*ChangeDocumentType Events 342
Syntax 343
Remarks 343
*Delete Events 343
Syntax 343
Remarks 343
Example 344
DocumentFolderSelected Event 344
Syntax 344
Remarks 344
*NewDocument Events 345
Syntax 345
Remarks 345
Example 346
OnProperties Event 346
Syntax 346
Remarks 346
*OpenInApplication Events 347
Syntax 347
PrepareCommand Event 348
Syntax 348
Remarks 348
*PrepareForOffline Events 348
Syntax 348
*Print Events 349
Syntax 349
Remarks 349
*Rename Events 349
Syntax 349
SelectTarget Event 350
Syntax 350

Remarks 350
*UpdateThumbnail Events 350
Syntax 350
DraftPrint_SetWatermark Event 351
Syntax 351
Remarks 351
Document Hybrid Events 352
Hybrid parts event sequence 352
AfterHybridPart Event 352
Syntax 352
BeforeHybridPart Event 352
Syntax 353
InitializeHybridParts Event 353
Syntax 353
BeforeConvert2Hybrid Event 353
Syntax 353
AfterConvert2Hybrid Event 354
Syntax 354
Document Project Copy Events 355
Document Project Copy Event Sequences 355
Create Project Copy event sequence 355
Confirm Merged with Master event sequence 356
Confirm Superseded by Master event sequence 356
Discard from Project event sequence 356
Link to Master event sequence 356
Release as Master Revision event sequence 357
AfterLinkToMaster Event 357
Syntax 357
Remarks 358
AfterMasterUpdate Event 358
Syntax 358
AfterTransferToNext Event 358
Syntax 358
Remarks 359
AfterUnlinkFromMaster Event 359
Syntax 359

Remarks 359
BeforeLinkToMaster Event 359
Syntax 359
Remarks 360
BeforeMasterUpdate Event 360
Syntax 360
BeforeTransferToNext Event 360
Syntax 360
Remarks 361
BeforeUnlinkFromMaster Event 361
Syntax 361
Remarks 361
*ConfirmMerged Events 361
Syntax 361
Remarks 362
*ConfirmSuperseded Events 362
Syntax 362
Remarks 362
*DiscardFromProject Events 362
Syntax 362
PreInitializeReleaseToMaster 363
Syntax 363
Remarks 363
PreBeforeReleaseToMaster 363
Syntax 363
Remarks 364
ProjectCopyExist Event 364
Syntax 364
Remarks 364
*ReleaseToMaster Events 365
Syntax 365
*RequireMerge Events 365
Syntax 365
Remarks 365
*UndoMakeObsolete Events 366
Syntax 366

Remarks 366
ValidateTargetFolder Event 366
Syntax 366
Remarks 367
Examples 367
Document Type Workflow Events 368
Document Type Workflow Event Sequences 368
Change To-Do Person event sequence 368
Change Workflow Manager event sequence 368
Execute transition event sequence 368
Revoke event sequence 369
*ChangeWFManager Events 369
Syntax 369
*ChangeWFPerson Events 369
Syntax 369
*ChangeWFState Events 370
Syntax 370
*RevokeWF Events 370
Syntax 370
Remarks 371
AfterReplaceReleased Event 371
Syntax 371
Remarks 371
Document Working Copy Events 373
Document Working Copy Event Sequences 373
Create Working Copy event sequence 373
Revoke Working Copy event sequence 373
Submit Working Copy event sequence 373
*CreateWC Events 374
Syntax 374
*RevokeWC Events 374
Syntax 374
*SubmitWC Events 374
Syntax 374
Folder Generic Events 376
Folder Generic Event Sequences 376

Change Folder Type event sequence 376
Delete Folder event sequence 376
New Folder event sequence 376
AfterNewFolder Event 377
Syntax 377
BeforeNewFolder Event 377
Syntax 377
*ChangeFolder Events 378
Syntax 378
FileUploaded Event 378
Syntax 378
Package Events 379
ExportPackage_ChangeState Event 379
Syntax 379
Remarks 379
Example 379
ImportPackage_AfterImportedFromPortal Event 380
Syntax 380
ImportPackage_AfterReadProperties Event 380
Syntax 380
ImportPackage_ChangeState Event 380
Syntax 380
Remarks 381
Example 381
Project Workflow Events 382
Project Workflow Event Sequences 382
Execute Transition event sequence 382
Reassign Managers event sequence 382
Reroute Project event sequence 383
*ChangeManager Events 383
Syntax 383
*ExecuteTransition Events 383
Syntax 383
*Reroute Events 384
Syntax 384
BeforeExpandItem Event 384

Syntax 384
Remarks 385
Example 385
InitializeExpandItem Event 387
Syntax 387
Remarks 387
InitializeWizard Event 387
Syntax 387
PrepareBrowser Event 388
Syntax 388
Remarks 388
Examples 389
TerminateWizard Event 389
Syntax 389
Remarks 390
Property Page Events 391
System Property Page Event Sequence 391
<PageName>Page_IsVisible Event 391
Syntax 391
Remarks 391
Example 392
<PageName>Page_CanEdit Event 392
Syntax 392
Remarks 392
<PageName>Page_CanApply Event 392
Syntax 392
Remarks 393
Example 393
Publishing Events 394
DocSync_Failed Event 394
Syntax 394
Remarks 394
DocSync_Options Event 394
Syntax 394
Remarks 394
DocSync_Succeeded Event 394

Syntax 394
Remarks 395
PublisherDestinationEvent_AfterPublish Event 395
Syntax 395
Example 395
PublisherDestinationEvent_HotspotUpdated Event 396
Syntax 396
PublisherSourceEvent_BeforePublish Event 396
Syntax 396
Return Value 397
Example 397
PublisherSourceEvent_AfterPublish Event 397
Syntax 397
Example 398
Vault Events 399
Vault Event Sequences 399
Change scope event sequence 399
Change view event sequence 399
Close event sequence 399
Open event sequence 399
ChangeScope Event 399
Syntax 399
Remarks 400
Example 400
ChangeView Event 401
Syntax 401
Remarks 401
Close Event 401
Syntax 401
Parameters 402
Remarks 402
NewProfile Event 402
Syntax 402
Parameters 402
Open Event 402
Syntax 402

Parameters 402
Remarks 402
Workflow Definition Events 403
Workflow Event Sequences 403
Execute transition event sequence 403
Reassign Managers event sequence 403
Reroute event sequence 404
*ExecuteTransition Events 404
Syntax 404
Remarks 405
*NewRevision Events 405
Syntax 405
*Reroute Events 406
Syntax 406
*Wizard Events 406
Syntax 406
Meridian API Constants 407
VBScript Examples 410
Validating unique property values 410
Clearing the property values of copied documents 410
Setting property values based on workflow transitions 411
Sending email messages 411
Automation Objects 413
Object Arguments 415
VBScript and PowerWeb 416
VBScript Functions Not Available in PowerWeb 416
Notes About Functionality 417
Formatting Text With RTF Codes 418
Confirmation Pages 421
Background 421
Simplified Script Sequence 422
Supported Events 423
Relevant Properties and Methods 425
Procedures 426
Add Confirmation Pages to VBScript 426
Add Confirmation Pages to Custom Command record 427

Meridian Enterprise Command Identifiers **428**
Identifiers 428
TagExtractor Component **430**
TagExtractor Object **431**
TagExtractor Object Methods 431
CreateHotSpotList Method 431
Syntax 431
Return Value 431
Remarks 431
Example 432
Open Method 432
Syntax 432
Return Value 432
Example 432
SetAdditionalFilter Method 432
Syntax 432
Return Value 433
Remarks 433
Examples 433
SyncTagReferences Method 433
Syntax 433
Return Value 433
Remarks 433
Example 433
SaveLogToFile Method 434
Syntax 434
Return Value 434
Remarks 434
Example 434
TagCollection Method 434
Syntax 434
Return Value 434
Remarks 435
Example 435
TagCollectionEx Method 435
Syntax 435

Return Value 435
Remarks 435
Example 436
TagCollection Object 437
TagCollection Object Properties 437
TagPrefix Property 437
Syntax 437
Remarks 437
TagPostfix Property 437
Syntax 437
Remarks 437
Tag Object 438
Tag Object Properties 438
Tag Property 438
Syntax 438
Example 438
TagType Property 438
Syntax 438
Example 438
BlockList Property 439
Syntax 439
Remarks 439
Example 439
ObjectList Property 439
Syntax 439
Remarks 439
Example 439
TagIterator Object 440
TagIterator Object Properties 440
IsValid Property 440
Syntax 440
Count Property 440
Syntax 440
Tag Property 440
Syntax 441
Remarks 441

TagType Property 441
Syntax 441
Remarks 441
BlockArray Property 441
Syntax 441
Remarks 441
ObjectArray Property 441
Syntax 441
Remarks 442
TagIterator Object Methods 443
Start Method 443
Syntax 443
Return Value 443
Remarks 443
Example 443
Step Method 443
Syntax 443
Return Value 444
Remarks 444
Example 444
TagExtractor Result Codes 445
Publishing And Rendering Options 446
Glossary 452
Index 468

Meridian Enterprise Introduction

The Meridian Enterprise product suite is an on-premises engineering document management system (EDMS) and asset lifecycle information management (ALIM) Accruent solution. Meridian Enterprise streamlines engineering-related business processes in specific industries including chemicals, pharmaceuticals, oil and gas, metals and mining, and utilities. Meridian helps your organization manage technical documents and drawings throughout its lifecycle, providing a centralized repository to store, organize, and control engineering documentation.

Meridian Enterprise includes:

- **PowerWeb**: Create and modify engineering data.
- **Explorer**: Search and retrieve engineering information.
- **Configurator**: Customize document types, workflow, roles, and other organizational procedures.
- **Administrator**: Manage users, licenses, and vaults.
- **Administration Console**: Manage users, licenses, jobs, repositories, and other settings.
- **Application Integration**: Streamline working with file-based applications by utilizing site cache and managing local workspace.
- **Microsoft Office link**: Synchronize metadata and references with Microsoft Office applications.
- **Autodesk links**: Synchronize metadata and references with AutoCAD family, Revit, and Inventor.
- **MicroStation link**: Synchronize metadata and references with MicroStation.

Meridian Enterprise supports regulatory compliance. it includes an FDA module to help your organization maintain the integrity and traceability of its documentation. The Meridian Asset Management module enhances, automates, and streamlines asset operations throughout their lifecycle by linking them with engineering content such as drawings and technical specifications.

Meridian Enterprise can integrate with Meridian Portal, a web application for internal and external collaboration in engineering projects. Contact your Accruent account representative for further information.

What's In This Guide

The primary method for customizing Meridian to meet your requirements, besides vault configuration described in the *Meridian Enterprise Configuration Guide*, is to use Visual Basic Scripting Edition (VBScript). VBScript is a subset of the Visual Basic programming language that is built into Meridian and many other applications. VBScript provides programmatic access to many Meridian event procedures, functions, and objects such as vaults, folders, documents, properties, and so on, so that you can automate simple tasks. The Meridian objects that are accessible by VBScript are described in Meridian Object Model.

VBScript is used by Meridian in two ways:

- Configuration expressions as described in Configuration Expressions
- Meridian event handling procedures as described in Understanding Meridian event procedures

This reference describes how to use VBScript with Meridian. For general information about the VBScript language, see the VBScript Language Reference web site.

> **Note:**
> - VBScript is not as complete a programming language as Visual Basic for Applications, which is built into Microsoft Office and other applications such as AutoCAD.
> - The examples shown in this documentation are for educational purposes only, may not represent a complete solution, and are not intended for use in a production environment.

Who Should Read This Guide

This guide is intended for advanced Meridian users who are responsible for vault configuration and security. You should be familiar with:

- The Windows® operating system, including its use in a network environment
- All aspects of using Meridian from an end user's perspective
- How your organization uses and manages documents as it conducts business
- The practices and requirements of user groups who will use the system
- Basic document control methodologies such as revision control
- Programming with VBScript

Technical Support

Technical support for Accruent products is available from a variety of sources if you have an active support contract. Your first source of support is the authorized contacts designated by your company to participate in the support contract. They are the persons that are responsible for resolving problems with Accruent software before contacting outside sources of support. If your company works with a Accruent Partner, that partner is your second source of support. Accruent Partners are responsible for providing technical support to their customers in order to maintain their status as Accruent Partners. Accruent will assist the partner company, if necessary, to help resolve your problem. If your company is a direct Accruent customer, your authorized contacts may communicate directly with Accruent to resolve your problem.

Accruent Partners and direct customers have access to all of these Accruent technical support resources:

- Support Cases: Around the clock support issue entry, update, and status
- Meridian knowledge base: Continuously updated problem solutions, minor releases, updates, and how-to articles about advanced techniques
- Email notifications: Immediate alerts to support issue status changes
- Telephone support: Direct access to highly qualified software support engineers with extensive experience in Accruent products

The available support contract options, terms, and other details are described in documents that are available from your Accruent Partner.

Meridian Enterprise Script Editor

The Meridian Enterprise Script Editor is a simple text editor for entering VBScript code. The code you create is saved in your Meridian vault configuration. Configuration expressions are saved as individual code blocks, and the Meridian event procedures code is saved as one code block.

The Meridian Enterprise Script Editor can help you to build correct VBScript code by listing all available objects, functions, and constants for easy selection. As you enter VBScript code in the code pane, the Meridian Enterprise Script Editor constantly monitors the syntax of your code. When the current code is correct, the status bar displays **Script is valid**. If the syntax of the current code is incorrect, the status bar displays the error and the location of the error so that you can easily correct it.

If you are entering a configuration expression, when you are finished entering code, you can test the code by clicking the **Evaluate the expression** button  . The Meridian Enterprise Script Editor will evaluate the expression and display the result so that you can confirm that the result is what you expect.

We include many examples of code that can be implemented in the VBScript Editor in our documentation. If you need assistance with your implementation, reach out to your Accruent representative.

Edit VBScript Code

Not all options described below will be available in every instance of the VBScript Editor. Learn more about the VBScript Editor user interface below.

To edit a code block:

1. Open the VBScript Editor.

   You can do this in one of three ways:

   - Click the **Edit Events** button  in the toolbar at the top of the Configurator screen.
   - Click the **Edit Events** button next to the place in the Configurator where the code block is applied.
   - Click the **Meridian Enterprise Script Editor** button  next to a place where configuration expressions can be applied.

2. To enable syntax coloring, select the **Use syntax coloring** check box.

   This check box appears at the bottom of the Script Editor pane. The performance of loading large scripts can be improved by disabling this option.

3. To enable the out-of-the-box script validation functionality, select the **Use validation** check box.

   This check box appears at the bottom of the Script Editor pane. When this check box is selected and the current code is correct, the status bar displays **Script is valid**. If the syntax of the current code is incorrect, the status bar displays the error and the location of the error so that you can easily correct it.

4. Use the editor to write your custom code.
   - To add an object to your code:
     a. Select the object from the **Object Browser** menu.

        The Script Editor enters the object name in the editing pane.

     b. To delineate a subclass of the current object, type a period after the object name.

        The Script Editor displays a popup list of the current object's properties and methods for you to select.

     c. If you want to use an object method:
        i. Select a method name from the list.
        ii. Type an open parenthesis.

            The editor shows a tooltip in the code pane of all required and optional parameters to assist you.

        iii. Provide values for any parameters that you want to use.
        iv. Ensure that all required parameters are specified.
        v. Type a close parentheses.
        vi. Repeat as necessary.
   - To add an event or function to your code, select it from the **Events and Procedures** menu.

   We include many examples of code that can be implemented in the VBScript Editor in our documentation. If you need assistance with your implementation, reach out to your Accruent representative.

5. To test a configuration expression, click the **Evaluate the expression** button  .

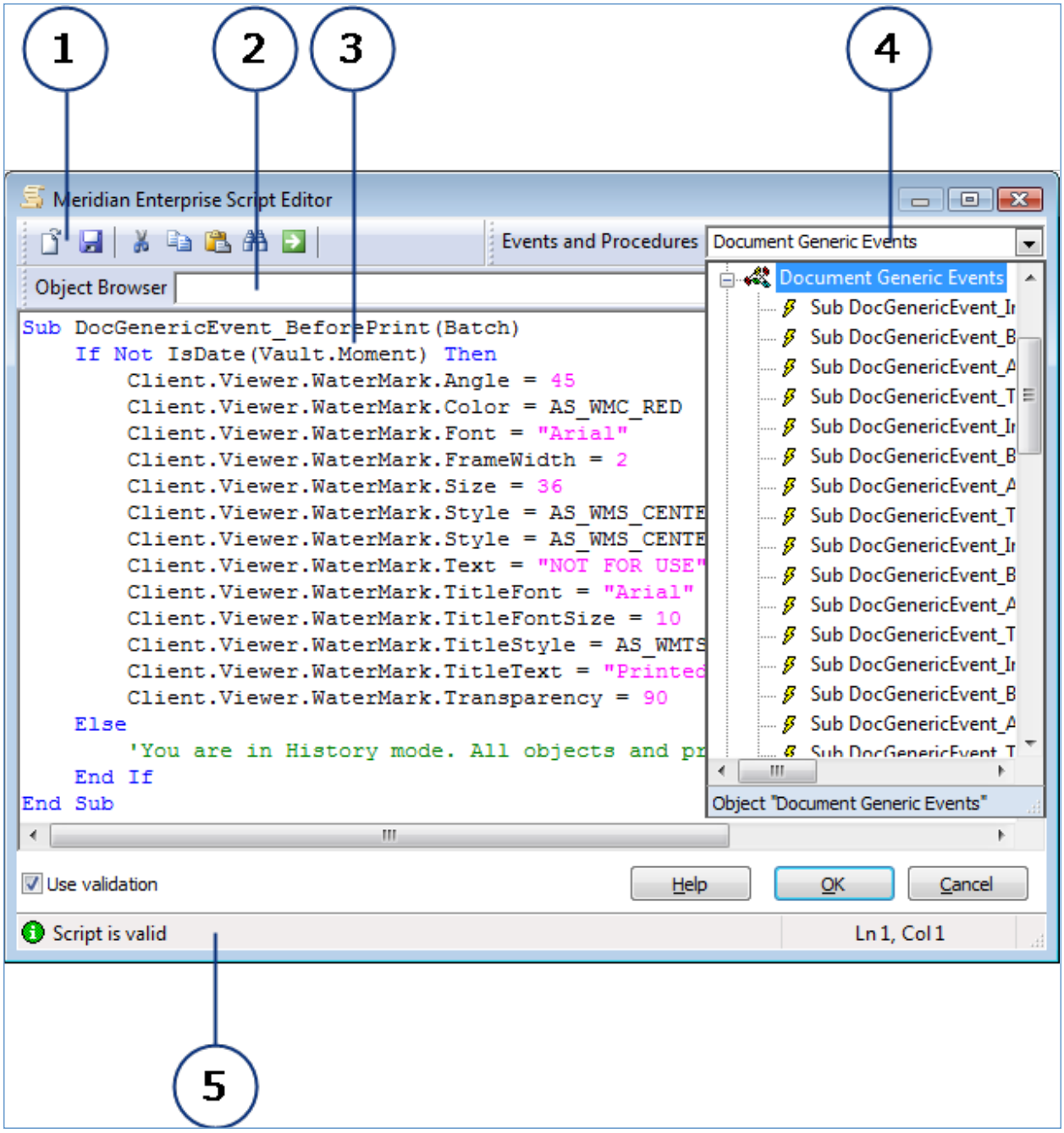
   This button only appears in the Script Editor for configuration expressions. Learn more about configuration expressions.

   The Meridian Enterprise Script Editor will evaluate the expression and display the result so that you can confirm that the result is what you expect.

6. Once you are satisfied with your code, click **OK**.

User Interface

The following diagram and tables are used to describe the VBScript Editor user interface.



The Meridian Enterprise Script Editor is composed of several elements described in the following table.

Features of the VBScript editor

Number	Name	Description
1	Toolbar	Contains buttons for transferring text with the Clipboard and searching code for text.
2	Object Browser	Displays a tree view of the VBScript built-in functions and constants, and the Meridian objects, properties, methods, functions, and constants. For information about the Meridian objects, see Meridian Object Model. For information about the Meridian functions, see Meridian functions.
3	Code pane	The area for entering VBScript code.
4	Events and Procedures	Displays a tree view of the Meridian events and any custom functions that have been defined. For information about the Meridian events, see Meridian event procedures. This view is only available when editing the Meridian events, as described in Create and Edit Event Procedures.
5	Status bar	Displays indicators for the current script's validity (if the **Use validation** option is enabled) and the current cursor position. The performance of loading large scripts can be improved by disabling **Use syntax coloring** option.

The commands for each icon on the toolbar are described in the following table.

Toolbar icons

Icon	Shortcut	Description
	Ctrl + O	Load a file from disk to completely replace the current code block. Available only when editing the Meridian events, as described in Create and Edit Event Procedures.
	Ctrl + S F12	Save the current contents of the editor to a file on disk. The default file name that is calculated includes the current date. Available only when editing the Meridian events, as described in Create and Edit Event Procedures.
	Ctrl + X	Cut the selected text to the Clipboard.
	Ctrl + C	Copy the selected text to the Clipboard.
	Ctrl + V	Paste text from the Clipboard.

Icon	Shortcut	Description
	Ctrl + F	Find specific text in the current code block. Available only when editing the Meridian events, as described in Create and Edit Event Procedures.
	Ctrl + G	Go to a specific line number in the current code block.
		Evaluates a configuration expression and displays the result so that you can confirm that the result is what you expect. This button only appears in the Script Editor for configuration expressions. Learn more about configuration expressions.

Effects Of Custom Scripting

Meridian Enterprise client performance can be affected by vault configuration options and custom scripting. However, the PowerWeb is must less sensitive than the PowerUser client.

Following are some areas in scripting and configuration that can directly affect the user experience.

- **Property events**

  These script events occur when a lookup list is generated or when a value has changed, for example. Extensive scripting of these events can result in slowness while editing properties. Avoid heavy actions like table queries on these events.

- **Custom command/page/action visibility events**

  These events calculate whether items should be available in the user interface. They can make switching documents and showing menus slow.

- **Before/After events**

  These events occur in the same transaction so the client is suspended during the operation. Be careful how much customization you do in these events because it can make an operation very slow.

- **Settings versus scripting versus tables**

  The **Vault.Option** property for retrieving custom settings that are stored in the vault configuration is faster than using a table query for constant values and is more flexible compared to scripting.

- **Security validation**

  Evaluating security privileges in scripting can be very slow. The **User.HasPrivilege** and **User.HasRole** properties can slow down events or operations.

- **CAD link property events**

  These occur for specific events like workflow changes. In most cases, scripting delays during these events are not critical unless they wait for a result.

- **Special cases**

  Some events, for example, related to creating a project copy or assembly, may not be supported in PowerWeb. This is because of the performance delays that they may cause because PowerWeb is not a fully interactive client.

To help with optimizing scripting and configuration, consider using script logging as described in Debugging VBScript.

Debugging VBScript

The Visual Studio programmer has the convenience of a debugger built into that development tool but beyond the script validation feature of the Meridian Enterprise Script Editor, you cannot debug VBScript in the editor. For real debugging, you need a different tool. The tool to use depends on the client application where you want to debug the script:

- To debug VBScript running in a PowerWeb session, the debugging tool must be installed on the computer running Internet Information Services and you must have administrative rights on the computer. The Meridian vault configuration is cached by your browser during each PowerWeb session. You must log off of PowerWeb to close the current session and then log back on again to load any script changes that you make.
- All other Windows applications where your VBScript code runs can be debugged using Visual Studio as described below.

> **Note:**
> Visual Studio 2008 or higher must be installed on the computer running the application from which you want to debug. Other debugging tools can be used as long as they support attaching themselves to a running process.

Debug in Visual Studio

To debug VBScript in any Windows application with Visual Studio:

1. Start the application that hosts the VBScript that you want to debug.
2. Perform the actions that lead up to the point where your VBScript is about to execute and you want to start debugging.
3. Start Visual Studio.

   The Visual Studio **Start Page** appears.
4. On the **File** menu, select **New Project**.

   The **New Project** dialog appears.
5. In **Project Types**:
   a. Expand **Other Project Types**.
   b. Click **Visual Studio Solutions**.
   c. Select **Blank Solution**.
   d. Click **OK**.

A new project is created.

The name of the project is irrelevant because you will not add any code to the project and will discard it when you are finished using the debugger.

6. On the **Tools** menu, select **Attach to Process**.

   The **Attach to Process** dialog appears.

7. Enable **Show processes from all sessions**.

   The **Available Processes** list refreshes to show additional processes.

8. In **Available Processes**, select the name of the executable for the host application that you started in step 1 and click **Attach**.

   For example, to debug a PowerWeb session:

   a. Select **w3wp.exe**.

   b. If the **Attach Security Warning** dialog box appears, click **Attach**.

      The **Solution Explorer** opens and lists one or more occurrences of **AMScriptHost.**

9. On the **Debug** menu, select **Break All** (or press Ctrl+Alt+Break).

10. Perform the action in the host application that will run the VBScript code that you want to debug.

    The Windows focus switches to Visual Studio and a new window opens, either **VBScript - script block [dynamic]** if a configuration expression is about to execute or **BlueCielo VB Script [dynamic]** if a sub-procedure or function in the event handler code module is about to execute. The next statement that will execute is highlighted and execution stops to allow you to debug.

    When script first executes after the process is attached to Visual Studio, the vault's active VBScript configuration expressions and the event handler code module are loaded in Visual Studio. You might have to step over (press F10) some configuration expressions or sub-procedures before the particular expression, sub-procedure, or function that you want to debug is entered. Whenever the event handler code module is loaded, you can set breakpoints in the **BlueCielo VB Script [dynamic]** window.

    To open the **BlueCielo VB Script [dynamic]** window:

    a. In the **Solution Explorer**, expand **Script Documents**.

    b. Expand **AMScriptHost**.

    c. Double-click **BlueCielo VB Script**.

    To make changes to script code, you must edit it in Meridian Enterprise Configurator, save your changes, reopen the vault in the host application, and restart this task from step 5.

11. When you are finished debugging, exit Visual Studio without saving the project.

Host Applications You Can Use to Debug

Following is a list of host applications related to Meridian Enterprise that you can attach to debug VBScript code:

- `Explorer.exe` to debug the Application Integration
- `Acad.exe` (or other application) to debug title block links
- `ConfiguratorU.exe` to debug the **Database Import Wizard**
- `PowerUserU.exe` to debug the **Document Import** tool
- `W3WP.exe` to debug PowerWeb on the web server

Enable VBScript Logging and Viewing

To further aid in debugging, you can enable logging of VBScript events, command scripts, evaluated expressions, and external function calls. You can enable logging only for PowerUser and PowerWeb.

To enable VBScript logging and viewing:

1. If PowerUser is open, close it.
2. In the Windows registry of the client PC, set the following registry values.

   These are also described in the *Registry Keys* section of the *Meridian Enterprise Administrator's Guide :*

   - **EnableScriptLog** in HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Client

     EnableScriptLog in `HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Client`
   - **File**, **MaxSize**, **Format**, **Filter**, and **Mode** in HKEY_CURRENT_USER\Software\Cyco\AutoManager Meridian\CurrentVersion\PowerUser\Settings\Log

     File, **MaxSize**, **Format**, **Filter**, and **Mode** in `HKEY_CURRENT_USER\Software\Cyco\AutoManager Meridian\CurrentVersion\PowerUser\Settings\Log`

The log file is restarted for each client session. The file includes the time, description, and source of the events.

> **Note:**
> - Set the preceding settings in the **Wow6432Node** branch in the registry for 32-bit clients.
> - Logging is not available for Application Integration or the application links.

- To add custom messages to the log, see Log method of the **Client** object.

Download Log with PowerWeb

To download the log with the PowerWeb:

1. Enable the **Download Log** menu command by creating or changing the **CanDownloadLog** setting to **1** in the user's PowerWeb profile file.

   This is described in *Deploy standard preferences* and *Edit PowerWeb User Profiles* in the *Meridian Enterprise Configuration Guide*.

   If the setting is **0** (default), the menu command is hidden.
2. Right-click to show the shortcut menu.
3. In the **Miscellaneous** group, click **Download Log**.

   The file downloads to your browser where you can save and view it.

Configuration Expressions

Depending on the configuration option where it is used, VBScript is available for configuration expressions to either calculate a value (file name, folder, property value, and so on) or to evaluate to **True** or **False** to apply an option (input required, field visibility, field read-only, and so on). You can use VBScript expressions in many places in a vault's configuration, including:

- Property pages
- Document naming
- Folder naming
- Lookup list tables and database queries
- Custom commands
- Workflow definitions
- Reports
- Document types
- Database Import Wizard

Each configuration expression is a single VBScript statement to be evaluated. Configuration expressions are limited in that they:

- Cannot contain more than one line of code, although they can call a predefined function that contains multiple lines of code.
- Cannot use variables.
- Should not execute other actions outside the scope of the current document, change property values, invoke long-running functions, or invoke any user interface functions. Such actions can have unexpected or unpredictable results.

You can use a VBScript expression anywhere you see the Meridian Enterprise Script Editor button  in Configurator. Clicking the button opens the Meridian Enterprise Script Editor where you can enter a VBScript expression.

For information about using the script editor, refer to *Meridian Enterprise Script Editor*.

Meridian Object Model

Meridian provides VBScript objects to represent most vault objects, such as documents, folders, users, and so on. These objects, in turn, provide related methods (actions) and properties (data) that you can use to manipulate vault objects. Many of the parameters of the Meridian event procedures are objects. You can view all of the Meridian objects, their methods, and properties with the **Object Browser** of the Meridian Enterprise Script Editor. Some of the Meridian object properties and methods use one or more predefined Meridian constants. The available Meridian constants can also be found in the **Object Browser** in the Meridian Enterprise Script Editor.

The entire collection of interrelated objects are known as an *object model*. The objects of the Meridian object model are described in the following topics.

Attachment Object

The **Attachment** object is a property of the **Attachments** object that represents an attachment of an email message.

Attachment Object Properties

The **Attachment** object provides the following properties.

FilePath Property

The path of the file to attach to the current message object.

Syntax

```
FilePath As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

Attachments Object

The **Attachments** object is a property of the **MailMessage** and **NewMailMessage** objects that is used to manage the file attachments of email messages.

Attachments Object Properties

The **Attachments** object provides the properties described below.

Attachment Property

An object that represents an attachment to the message. Read-only.

Syntax

```
Attachment(Index) As IASMailAttachment
```

Count Property

Total number of attachments to the message. Read-only.

Syntax

```
Count As Long
```

Attachments Object Methods

The **Attachments** object provides the methods described below.

Add Method

Adds an attachment to the current **NewMailMessage** object.

Syntax

```
Add(FilePath As String, [Name As String])
```

Parameters

Name	Description
FilePath	The path of the file to attach to the current **NewMailMessage** object.
Name	Optional name of the attachment.

Return Value

An **Attachment** object.

Remove Method

Removes an attachment from the current **NewMailMessage** object.

Syntax

```
Remove(Index)
```

Parameters

Name	Description
Index	A specific item in the attachments collection. Read-only.

RemoveAll Method

Removes all attachments from the current **NewMailMessage** object.

Syntax

```
RemoveAll()
```

Parameters

This method accepts no parameters.

Batch Object

The **Batch** object is passed as an argument to all event handlers. The same object is available for each event in a batch operation.

> **Note:**
> - A single-document operation is treated as a batch with size 1.
> - The **Batch** object is not supported by the CAD application links in remote mode.

Batch Object Properties

The **Batch** object provides the following properties, most of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

Argument Property

A variable stored for the **Batch** object during the current batch operation. Arguments can be used to pass parameters between procedures within the same batch operation.

Syntax

`Argument(`*`Name`*`) As Variant`

Following are the predefined, read-only **Batch** object arguments that you can use to obtain additional information that may be useful when working with particular events.

Batch object arguments

Argument	Description
Batch.Argument("__ $$RelatedProjectCopy")	A reference to the project copy document that is being released as a master document. Useful in the **DocGenericEvent_ BeforeNewDocument** event to move the master document to the correct location if it is being created for the first time.
Batch.Argument("__ $$RelatedVaultDocument")	A reference to a vault document that corresponds to briefcase file. The reference can be a ID string, **Document** object, or **AMDocument** object. Useful in the **BrcEvent_BeforeImport** event to relate a vault document to an incoming file.

Argument	Description
Batch.Argument ("DocumentPath")	Provides the location of the document in the local workspace starting from the root of the vault. This event will only process if the **NewLibraryComponentDocType** setting is disabled. This setting is described in the HKEY_CURRENT_USER\Software\Cyco\AutoManager Meridian\CurrentVersion\InventorLink article in the *Meridian Enterprise Administrator's Guide*.
Batch.Argument ("MatchedTypes")	Provides a semicolon-delimited list of internal type names matched to the document extension. This event will only process if the **NewLibraryComponentDocType** setting is disabled. This setting is described in the HKEY_CURRENT_USER\Software\Cyco\AutoManager Meridian\CurrentVersion\InventorLink article in the *Meridian Enterprise Administrator's Guide*.

Remarks

The value of **Name** is case-sensitive.

For more information about arguments, see Object Arguments.

ApplyToAll Property

Indicates whether a user selected the **Apply for all other documents of this type** option that is shown on wizard pages for batches of documents during custom command execution. The default is **True**.

Syntax

```
ApplyToAll As Boolean
```

Remarks

This property can be set in the ***<CommandName>*_Initialize** event using the constant **AS_CONFIRM_APPLY_FOR_ALL** to specify whether the default behavior should be to apply a stored value to all documents in the batch. The property can then be tested in the ***<CommandName>*_BeforeWizard** and ***<CommandName>*_AfterWizard** events to determine if the wizard was shown, the user changed the option, and if the properties on the wizard page have been applied to the first document in the batch and will be applied to the next document in the batch.

In PowerWeb, the same result is achieved by setting `Batch.ApplyToAll` to `True` in the relevant Initialize event the same result is achieved in PowerWeb. For example:

```
Sub DocCopyMoveEvent_InitializeCopy(Batch, TargetFolder)
    Batch.ApplyToAll = True  'Apply to all option in wizard default on
End Sub
```

This works in all cases except Single Document Workflow (DocWorkflowEvent_ InitializeChangeWFState) and Configurable Workflow (DocCWFEvent_InitializeExecuteTransition).

BatchIndex Property

Index of the current item in a batch operation. Starts at 1. Read-only.

Syntax

```
BatchIndex As Long
```

BatchSize Property

Number of items in the current batch operation. Returns 1 for non-batch operations. Read-only.

Syntax

```
BatchSize As Long
```

ConfirmationTitle Property

The title of the current confirmation dialog box.

Syntax

```
ConfirmationTitle As String
```

Example

See Confirmation method

CountFailed Property

The number of operations in the current batch that failed. Read-only.

Syntax

```
CountFailed As Long
```

CountSucceeded Property

The number of operations in the current batch that succeeded. Read-only.

Syntax

```
CountSucceeded As Long
```

IsFirstInBatch Property

True only for the first item in a batch operation. **True** for single documents. Read-only.

Syntax

```
IsFirstInBatch As Boolean
```

IsLastInBatch Property

True only for the last item in a batch operation. **True** for single documents. Read-only.

Syntax

```
IsLastInBatch As Boolean
```

ProcessAsBatch Property

Set to **True** to indicate that the current (custom) command acts as a batch operation upon the members of the **Batch** object. Set to **False** (default) to act only upon the selected document.

> **Note:**
> Set this property to **True** in the **DocGenericEvent_PrepareCommand** event before programmatically adding documents with the **Batch.AddDocuments** method in a custom command's **Initialize** event.

Syntax

```
ProcessAsBatch As Boolean
```

Batch Object Methods

The **Batch** object methods affect the current batch operation. They are described below.

Abort Method

Abort the current batch operation.

Syntax

```
Abort([Message])
```

Parameters

Name	Description
Message	Show the specified message to the current user.

Remarks

Uncommitted items in the batch are revoked.

AddDocuments Method

Adds the members of the specified collection to the **Batch** object.

Syntax

```
AddDocuments(Collection As Object)
```

Parameters

Name	Description
Collection	A collection of documents to add to the **Batch** object. Can be the result of **Vault.FindDocuments** or **Document.GetReferences**

Remarks

Not available in PowerWeb. This method is intended to be used in a custom command's **Initialize** event to add documents to a batch for processing.

AddDocumentToBatch Method

Adds a specified document to the **Batch** object.

Syntax

```
AddDocumentToBatch(DocID As String)
```

Parameters

Name	Description
DocID	The **DocID** value of the document to add.

Remarks

Not available in PowerWeb. This method is intended to be used in a custom command's **Initialize** event to add documents to a batch for processing.

AskConfirmation Method

Shows a confirmation prompt in the batch progress dialog with **Yes** and **No** radio buttons for answer selection and an optional comment field.

Syntax

```
AskConfirmation(ID As String, Label As String, [Default as AS_
CONFIRMATION_DEFAULT_VALUE], [Comment As AS_CONFIRMATION_COMMENT_
FLAGS])
```

Parameters

Name	Description
ID	Identifier for this confirmation prompt that can be used to refer to the user's answer.
Label	Text of the prompt.

Name	Description
Default	Default radio button.
Comment	Combination of flag values that determine whether comments are accepted and are mandatory or optional.

Example

See Confirmation method

AskInput Method

Shows an input field in the batch progress dialog.

Syntax

```
AskInput(ID As String, Label As String, [Type as AS_INPUT_RESULT_
TYPE], [Rule As AS_INPUT_VALIDATION_RULE], [Prototype As String])
```

Parameters

Name	Description
ID	Identifier for this input field that can be used to refer to the user's answer.
Label	Text of the field label.
Type	Optional value that represents the type of input that is accepted.
Rule	Optional value that represents the type of input validation performed.
Prototype	Optional name of an existing custom property from which to inherit the property's validation rules.

Example

See Confirmation method

Confirmation Method

Shows a custom confirmation dialog. This method should not be confused with the Confirmation Property.

Syntax

```
Confirmation(ID As String)
```

Parameters

Name	Description
ID	Identifier of the content to show in the dialog.

Return Value

Object with two properties, the user's input in **Value** and the user's comment (if any) in **Comment**.

Remarks

Confirmation dialog boxes are not supported if the **ActiveX compatibility mode** option is enabled as described in the *Personal Preferences* article in the *Meridian Enterprise User's Guide*. The available constants can be found in the **Constants** branch of the **Object Browser** in the Meridian Enterprise Script Editor.

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

FailCurrent Method

Fail the current item in the current batch operation.

Syntax

```
FailCurrent([Message])
```

Parameters

Name	Description
Message	Append the specified message to the **Comments** property.

Remarks

> **Important!**
> Use this method very carefully in managed change and Meridian Advanced Project Workflow operations, batch operations, or if references are involved. It could cause unexpected results.
>
> For example, if a project copy is made of a master document that has a reference to another document and the operation is canceled by validation in script, the referenced document might still be copied but without the reference (because the master document was not copied). If the successful project copy is then released as a master revision, the existing reference between the source documents will be deleted (because the project copy had no reference to the master document).
>
> In such scenarios, the validation script should produce a different result than aborting the operation, such as setting the target folder to **Nothing**, which prevents copying the referenced document.

Input Method

Shows the specified confirmation content.

Syntax

```
Input(ID As String) As Object
```

Parameters

Name	Description
ID	Identifier of the content to show in the dialog box.

Return Value

Object with two properties, the user's input in **Value** and the user's comment (if any) in **Comment**.

Example

See Confirmation method

PrintDetails Method

Shows a string in the batch progress dialog box.

Syntax

`PrintDetails(`*`String`* `As String)`

Parameters

Name	Description
String	The text string to show in the progress dialog box.

Remarks

To add a string to a the **Comment** log of a document, use the **Document.Log** method instead as described in Log method.

> **Note:**
> In Meridian 2021, event scripts such as this one do not work where focus is set on the folder level and a create a new document command for document type with wizard page (without template) is used.

```
1 Sub DocGenericEvent_BeforeNewDocument(Batch, Action, SourceFile, DocType, DocTemplate)
2 batch.PrintDetails "BeforeNewDocument"'(Document.FileName) 'Add your code here
3 End Sub
```

> In this scenario, the document is created, but the focus remains on folder level and the navigation view is not refreshed. This will be fixed in a future release.

ShowInfo Method

Shows one or more lines of text with an optional heading in a confirmation dialog. The first 5 lines of text are visible by default with a vertical scroll bar if more lines exist.

Syntax

`ShowInfo(`*`Info`* `As String,` *`Heading`* `As String)`

Parameters

Name	Description
Info	Text of the information.
Heading	Text of the heading.

Example

See Confirmation method

BCPropStorage Object

The **BCPropStorage** object represents the property information that was read by the DocCADLink_AfterReadMTBProperties event or that will be written by the DocCADLink_BeforeWriteMTBProperties event.

BCPropStorage Object Properties

The **BCPropStorage** object provides the following properties, all of which are read-only.

LayoutsNames Property

Gets the names of the page layouts in the current document.

Syntax

```
LayoutsNames As String
```

Returns

An array of names as strings.

Remarks

Use this property to retrieve the layout names to present to the user for selecting the layouts to render with the **MeridianQueue.RegisterDocument** property. Pass the selected layout names in the **publishOptions** parameter using the **_RENDERLAYOUTS_** option described in Publishing And Rendering Options.

This property can also be used to access the data in multiple title blocks defined in a single drawing. For an example of usage, see DocCADLink_AfterReadMTBProperties event.

This property is intended for AutoCAD and Revit drawings only. If the document does not contain multiple title blocks or sheet properties with the names configured in the application link (**AutoCAD** or **Revit** tabs, respectively) settings of the vault configuration, the returned array is empty.

BCPropStorage Object Methods

The **BCPropStorage** object methods affect the current **BCPropStorage** object. They are described below.

GetColumnDefs Method

Returns a collection of the specified data columns from the data table shown on the **Title Blocks** page by the CAD link after reading a title block from a drawing layout.

Syntax

```
GetColumnDefs(CollectionType As Integer)
```

Parameters

Name	Description
CollectionType	One of the following numeric values: **1**: Only the mandatory layout and block name columns **2**: Only the columns mapped in the link configuration **3**: All columns

Return Value

A collection of column definition objects. Each object has the following properties:

- **Name**: Internal name of the column
- **DispName**: Display name of the column
- **PropType**: One of the following property value types:
  - **0**: String
  - **1**: Integer
  - **2**: Date
  - **3**: Real number
  - **4**: Boolean
- **Direction**: One of the following property update directions as specified by the CAD link configuration:
  - **0**: Any direction
  - **1**: From Meridian to the document
  - **2**: From the document to Meridian
- **Hidden**: Indicates whether the column is hidden in the table
- **Order**: Column display order in the table

Remarks

Use this method to access the data in multiple title blocks defined in a single drawing. For an example of usage, see DocCADLink_AfterReadMTBProperties event.

Property Method

Returns an object that represents data stored in the title block data table for the specified property.

Syntax

```
Property(LayoutName As String, ColumnName As String)
```

Parameters

Name	Description
LayoutName	Name of layout that contains a linked title block with the specified column name.
ColumnName	Internal name of the column in the data table shown on the **Title Blocks** page by the CAD link for the specified drawing layout.

Return Value

The returned object has the following properties:

- **Value**: The read-write value of the specified property
- **ColumnDef**: Column definition object for the column that contains the value as described in GetColumnDefs method

Remarks

Use this method to access the data in multiple title blocks defined in a single drawing and to set attribute values (optional). For an example of usage, see DocCADLink_AfterReadMTBProperties event.

Client Object

The **Client** object represents the current client application (for example, PowerUser).

Client Object Properties

The **Client** object provides the following properties.

Argument Property

A variable stored with the **Client** object for as long as the current client session is active. Arguments can be shared by all instances of the client application on the same computer.

Syntax

```
Argument(Name) As Variant
```

Remarks

The value of **Name** is case-sensitive.

Following are the predefined, read-only **Client** object arguments that are set by MS Outlook and IBM Lotus Notes application links when email messages are stored in a vault. You can use these arguments to obtain additional information that may be useful when working with particular events. The value of each argument is evident from the argument name.

- AttachmentName
- AttachmentsCount
- BCC
- BodyFormat
- CC
- ConversationIndex
- ConversationTopic
- CRC32
- CreatingTime
- EntryID
- Importance
- MAPIFolder

- ReadReceiptRequested
- ReceivedByName
- ReceivedTime
- Saved
- SenderEmailAddress
- SenderName
- Sent
- SentOn
- Size
- Subject
- To
- UnRead

For more information about arguments, see Object Arguments.

BuildNr Property

The version number of the AmEdmUI.dll client library file.

Syntax

```
BuildNr As String
```

Remarks

Returns the same number regardless of the active client application. This property can be used to detect outdated client software installations.

ClientID Property

A number that represents the active client applications for the current transaction. The number is the sum of one or more **AS_CLIENTID** constants.

Syntax

```
ClientID As Long
```

ClientProductCode Property

Number that represents the client product. Read-only.

Syntax

```
ClientProductCode As String
```

Confirmation Property

Set this property to **True** or **False** to control the display of confirmation prompts that appear to users for some commands. **Action** is specified as one of the **AS_CONFIRM_ACTION** constants that corresponds to the prompt that you want to enable or disable, respectively. The available constants can be found in the **Constants** branch of the **Object Browser** in the Meridian Enterprise Script Editor.

Syntax

```
Confirmation(Action) As Boolean
```

Remarks

In PowerWeb, client confirmation is supported for these actions:

- **AS_CONFIRM_COPYFPRSUBLEVELS**: Defines what to do with FPR properties during project copy creation.
- **AS_CONFIRM_ALLOW_SKIPWIZARD_ALL**: Partly supported for workflow transition wizards. Not supported for new document/folder wizards and auto command wizards.
- **AS_CONFIRM_APPLY_FOR_ALL**: Partly supported for workflow transition wizards. Not supported for new document/folder wizards and auto command wizards.
- **AS_CONFIRM_COPY_FOLDER**: Use the DocGenericEvent_SelectTarget event.
- **AS_CONFIRM_IMPORT_FOLDER**: Use the DocGenericEvent_SelectTarget event.
- **AS_CONFIRM_PROJECT_FOLDER**: Use the DocGenericEvent_SelectTarget event.

CurrentScope Property

Name of the current scope.

Syntax

```
CurrentScope As String
```

CurrentView Property

Name of the active navigation view. Read-only.

Syntax

```
CurrentView As String
```

Remarks

This property is not supported in PowerWeb.

ImportDetails Property

Long integer that represents one or more **AS_IMPORT_DETAILS** constants. This property is set only when the **ImportType** property contains a valid value. Read-only.

Syntax

```
ImportDetails As AS_IMPORT_DETAILS
```

ImportType Property

Long integer that represents one or more **AS_IMPORTTYPE** constants. Available only for new documents. Read-only.

Syntax

```
ImportType As AS_IMPORTTYPE
```

Type Property

The client application platform. Returns **HTML** for PowerWeb. Read-only.

Syntax

```
Type As String
```

Viewer Property

See Viewer Object. Read-only.

Syntax

```
Viewer As IASViewer
```

Client Object Methods

The **Client** object methods affect the current client application. The client application executes these methods only after the current operation (and all its event procedures) have finished. These commands may invoke new events.

The **Client** object methods are described below.

GoTo Method

Select the specified object. Makes the specified object the current object.

Syntax

`GoTo(`*`Document/Folder`*`)`

Parameters

Name	Description
Document/Folder	**Document** or **Folder** object to make the selected object.

Remarks

The method is available in PowerWeb with these limitations:

- If the target document or folder is not visible in the current Scope, the `Client.Goto` statement is ignored.
- If the target document or folder is not visible in the view, PowerWeb switches to the Explorer view.
- In ActiveX mode, `Client.Goto` is ignored.

GoToView Method

Shows the specified view.

Syntax

`GoToView(`*`ViewName`*`)`

Parameters

Name	Description
ViewName	Name of the Navigation view to make the current view.

Remarks

Not available in PowerWeb.

Log Method

Sends a message to the VBScript debugging log.

Syntax

```
Log(Flags As AS_LOG_FLAGS, Message As String, [Source As String])
```

Parameters

Name	Description
Flags	Long integer that represents one or more of the **AS_LOG_FLAGS** constants. You define the meaning of each of the **AS_LOG_FLAGS** constants. They are only provided as a convenience for filtering the log and have no other effect than the icons shown for the log entries. Use **AS_LF_DEBUG** to limit logging to debug mode. Use **AS_LF_MASK** to combine constants.
Message	Text to send to the log.
Source	Optional name of the source of the message.

Remarks

Not available in PowerWeb. To configure and view the log, see Debugging VBScript.

NewMailMessage Method

Creates a new email message object.

Syntax

```
NewMailMessage() As IASMailMessage
```

Parameters

This method accepts no parameters.

Return Value

A new **MailMessage** object as described in MailMessage Object.

Remarks

Not available in PowerWeb.

OpenInApplication Method

Opens the specified document in the application that is registered in Windows for the document's file extension.

Syntax

```
OpenInApplication(Document)
```

Parameters

Name	Description
Document	An object that represents the document to open.

Remarks

Not available in PowerWeb.

Refresh Method

Refreshes the client application.

Syntax

```
Refresh(Flags As AS_REFRESHFLAG)
```

Parameters

Name	Description
Flags	Long integer that represents one or more of the **AS_REFRESHFLAGS** constants.

Remarks

Not available in PowerWeb.

Document Object

The **Document** object refers to the selected document and is available to all event procedures that apply to documents. The **Document** object is not passed as a parameter to most event procedures.

Document Object Properties

The **Document** object provides the following properties, most of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

BriefCase Property

An object that represents a briefcase. If the parent object is a document, this is the briefcase to which the document is locked. If the parent object is a report (transmittal sheet), this is the briefcase that contains the report. Read-only.

Syntax

```
BriefCase As IASBriefCase
```

CanEditProperty Property

True if the user can edit this property, **False** if editing is not allowed. Read-only.

Syntax

```
CanEditProperty As Boolean
```

ConcurrentEngineeringRuleProperty Property

Specifies the way in which documents are locked when waiting lists are in use for project copies. The constant `AS_CE_Rule` contains enumerations of behaviors that you can specify for a project copy:

- `AS_CER_DEFAULT`: Default behavior
- `AS_CER_MERGE_WF`: Current behavior
- `AS_CER_NOT_ALLOWED`: Allow only one project copy
- `AS_CER_SERIAL_WITH_RELEASE`: Allow multiple project copies
- `AS_CER_SERIAL_WITH_TRANSFER`: Reserved for future use

If you want to change the concurrent engineering rule for a document that is in progress using script, you must add the **ConcurrentEngineeringRule** property to the **SafeProperties** registry setting. This setting is described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager*

Meridian\CurrentVersion\AMOMLUI\SafeProperties article in the *Meridian Enterprise Administrator's Guide*.

Syntax

```
CanEditProperty As AS_CE_RULE
```

Created Property

The date and time when the document was created. Read-only.

Syntax

```
Created As Date
```

CreatedBy Property

An object that represents the person who created the document. Read-only.

Syntax

```
CreatedBy As IASUser
```

CWFManagers Property

The names of the users assigned as the workflow definition managers of the document. Read-only.

Syntax

```
CWFManagers As String
```

CWFState Property

An object that represents the current workflow definition state of the document. Read-only.

Syntax

```
CWFState As IASWorkflowState
```

CWFTodoPersons Property

The names of the users assigned as the to-do persons of the document. Read-only.

Syntax

```
CWFTodoPersons As String
```

DocumentType Property

An object that represents the document type of the document. Read-only.

Syntax

```
DocumentType As IASDocumentType
```

FileName Property

The file name (**Display Name** property) of the document or briefcase.

Syntax

```
FileName As String
```

Remarks

For events to occur when setting this property for a document, the **RenameEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

FileType Property

The file type (as registered in Windows on the current computer) of the document. Read-only.

Syntax

```
FileType As String
```

ForgeURL property

The URL to view the current document in Autodesk Forge. Can be modified with ChangeForgeURL Method.

Syntax

```
ForgeURL As String
```

GlobalID Property

The ID of the document that is unique among all vaults. Read-only.

Syntax

```
GlobalID As String
```

HasIncomingReferences Property

True if the document has incoming Meridian references. Read-only.

Syntax

```
HasIncomingReferences As Boolean
```

HasOutgoingReferences Property

True if the document has outgoing Meridian references. Read-only.

Syntax

```
HasOutgoingReferences As Boolean
```

HasRedlines Property

True if the document has Meridian redlines attached to it. Read-only.

Syntax

```
HasRedlines As Boolean
```

HasRendition Property

This property is now obsolete. Use RenditionStatus Property instead.

HasRenditionRedlines Property

True if the rendition of the document has Meridian redlines attached to it. Read-only.

Syntax

```
HasRenditionRedlines As Boolean
```

HybridParts Property

A list of the hybrid part document names.

Syntax

```
HybridParts() As Array
```

ID Property

Returns the ID of the current object. The value is unique within the current vault. For the ID value that is unique among all vaults, retrieve the **GlobalID** property value, if available. Read-only.

Syntax

```
ID As String
```

ImportType Property

Long integer that represents one or more **AS_IMPORTTYPE** constants. Available only for new documents. Read-only.

Syntax

```
ImportType As AS_IMPORTTYPE
```

IsChangedByPortal Property

True if the document returned from Meridian Portal is changed. Read-only.

Syntax

```
IsChangedByPortal As Boolean
```

Example

This sample code releases a quick change if the document was not changed in Meridian Portal.

```
Sub ImportPackage_ChangeState (package, previousState, User)
    Dim Doc, arrDoc, i
    If package.Status = AS_IPS_Imported Then
            arrDoc = Package.GetDocuments()
        For i = 0 To Ubound(arrDoc)
             Set Doc = Vault.GetDocument(arrDoc(i))
            If Doc.Property("AMDocumentPropertySet._ISCHANGEDBYPORTAL") = False Then
                ' revoke quick change
                Doc.RevokeChange
            End If
           Next
    End If
End Sub
```

IsUnderRevision Property

True if a working copy exists for the current document. Read-only.

Syntax

```
IsUnderRevision As Boolean
```

IsUniqueValue Property

True if the specified value for the specified document property is unique within the current vault. Read-only.

Syntax

```
IsUniqueValue(PropertyName, Value) As Boolean
```

LayerTranslationTable Property

Layer translation table to use for translating the document layers when generating a PDF rendition from the document. Translation tables are created by your administrator in the Meridian Enterprise Server Administration Console.

Syntax

```
LayerTranslationTable As String
```

LayoutsNames Property

Gets the names of the page layouts in the current document.

Syntax

```
LayoutsNames As String
```

Returns

An array of names as strings.

Remarks

Use this property to retrieve the layout names to present to the user for selecting the layouts to render with the **MeridianQueue.RegisterDocument** property. Pass the selected layout names in the **publishOptions** parameter using the **_RENDERLAYOUTS_** option described in Publishing And Rendering Options.

This property can also be used to access the data in multiple title blocks defined in a single drawing. For an example of usage, see DocCADLink_AfterReadMTBProperties event.

This property is intended for AutoCAD and Revit drawings only. If the document does not contain multiple title blocks or sheet properties with the names configured in the application link (**AutoCAD** or **Revit** tabs, respectively) settings of the vault configuration, the returned array is empty.

LockingProjectCopy Property

A **Document** object that represents the project copy that is locking the selected document. Read-only.

Syntax

```
LockingProjectCopy As IASDocument5
```

MasterDocument Property

A **Document** object that represents the master document from which the current project copy document was copied. Read-only.

Syntax

```
MasterDocument As IASDocument5
```

Modified Property

The date and time when the document content was last modified. Read-only.

Syntax

```
Modified As Date
```

ModifiedBy Property

An object that represents the person who last modified the document content. Read-only.

Syntax

```
ModifiedBy As IASUser
```

ParentFolder Property

An object that represents the parent folder of the document. Read-only.

Syntax

```
ParentFolder As IASFolder3
```

ParentProject Property

A **Folder** object that represents the parent project that contains the current project copy document. Read-only.

Syntax

```
ParentProject As IASFolder3
```

Path Property

The relative path of the document from the root of the vault. Read-only.

Syntax

```
Path As String
```

ProjectCopy Property

An object that represents a project copy of the selected document. Read-only.

Syntax

```
ProjectCopy As IASProjectCopy
```

Property Property

Gets or sets the value of the specified (String) document property.

Syntax

```
Property(Name) As Variant
```

Rendition Property

The vault or Local Workspace path of the document and its rendition. Read-only.

Syntax

```
Rendition As String
```

RenditionStatus Property

The status of the current rendition stored in the vault. The property contains a long integer that represents one or more **AS_RENDITION_STATUS** constants. Read-only.

Syntax

```
RenditionStatus() As AS_RENDITION_STATUS
```

Remarks

When the content of the source document is replaced by a user, this property returns **AS_RS_OUTDATED**.

Revision Property

The revision number of the document.

Syntax

```
Revision As String
```

ShareID Property

A server ShareID to get a transaction object from another process. Read-only.

Syntax

```
ShareID As String
```

Size Property

The file size of the document content. Read-only.

Syntax

```
Size As Long
```

StatusText Property

The text of the document's current workflow **Status** property.

Syntax

```
StatusText As String
```

UnderRevisionBy Property

An object that represents the user who owns the working copy of the document. Read-only.

Syntax

```
UnderRevisionBy As IASUser
```

VersionID Property

The ID of the current revision of the document that is unique within the current vault. Read-only.

Syntax

```
VersionID As String
```

WorkFlowAction Property

The action that the current to-do person is assigned to perform on the document. Read-only.

Syntax

```
WorkFlowAction As String
```

WorkFlowManager Property

An object that represents the user assigned to manage the current document type workflow of the document. Read-only.

Syntax

```
WorkFlowManager As IASUser
```

WorkflowState Property

An object that represents the current workflow state of the current workflow object. Read-only.

Syntax

```
WorkflowState As IASWorkflowState
```

WorkFlowTodoPerson Property

An object that represents the user assigned as the to-do person of the document in the current document type workflow state. Read-only.

Syntax

```
WorkFlowTodoPerson As IASUser
```

Document Object Methods

The **Document** object methods affect the current document object.

AddRendition Method

Adds rendition content from the specified file to the current document.

Syntax

```
Sub AddRendition (File As String, [NewRevision As Boolean = True])
```

Parameters

Name	Description
File	Path to a file outside the vault to import as rendition content for the current document.
NewRevision	If **True** (default), creates a new revision of the rendition.

Remarks

Rendition revisions are separate from the parent document revisions. They need not be synchronized and there may be many of one related to one of the other.

AddReorderingComment Method

Adds a reordering comment to the Project Copy.

Syntax

```
AddReorderingComment(reorderingComment as String)
```

Parameters

Name	Description
reorderingComment	A string containing the comment to be added to the project copy.

ApplyPropertyValues method

Saves the current property values to the current document object.

Syntax

```
ApplyPropertyValues()
```

Parameters

This method accepts no parameters.

AttachWaitingProjectCopy Method

Attaches an existing document as a waiting Project Copy. This updates waiting priority and retains the creation date of the existing document. You can also use the AddReorderingComment method to add a reordering comment to the Project Copy.

Syntax

```
AttachWaitingProjectCopy(ProjectCopy as Object) As IASDocument15
```

Parameters

Name	Description
ProjectCopy	An object representing the project copy to which the document is to be attached.

CalculateFileType method

Resets the **FileType** property of the current document object.

Syntax

```
CalculateFileType()
```

Parameters

This method accepts no parameters.

CallRemote Method

Executes a remote procedure call.

Syntax

```
CallRemote(URL As String, Username As String, _
       Password As String, RemoteVault As String, _
       Script As String, [Args], [Flags As Long = 2])
```

Parameters

Name	Description
URL	Location of the server with which to connect.
Username	User account to use to connect to the server.
Password	Password of the user account.
RemoteVault	Name of the remote vault with which to connect.
Script	Name of the procedure to execute on the remote server.
Args	Optional arguments for the procedure to execute.
Flags	Optional flags from the **AS_CALLREMOTE_FLAGSS** enumeration.

Example

The following examples show how to use this method.

Example function defined in the called vault:

```
Function RemoteTest (First, Second, Third)
  RemoteTest = "RemoteTest returns: " & First & ", " & Second & ", "
& Third
End Function
```

Example procedure defined in the calling vault:

```
Sub Test_Execute(Batch)
  vArg = Array ("One ", "Two", "Three")
  vRes = Vault.CallRemote ("http://MyServer/Meridian", "MyUserName",
_
         "MyPassword", "MyVaultName", "RemoteTest", VArg, AS_CRF_
MULTIARGS)
  WinMsgBox vRes
End Sub
```

ChangeDocumentType Method

Changes the document type of the current document object.

Syntax

`ChangeDocumentType(`*`DocType`*`)`

Parameters

Name	Description
DocType	An object that represents the new document type

Remarks

For events to occur for this method, the **ChangeDocTypeEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**(default).

ChangeForgeURL Method

If a rendition is already available in Autodesk Forge, changes the ForgeURL property of the document based on the parts that you specify.

Syntax

`ChangeForgeURL(`*`ForgeURL1`*`, `*`ForgeURL2`*`)`

Parameters

Name	Description
ForgeURL1	A string that represents the first part of the Forge URL for the document, for example, the base URL.
ForgeURL2	A string that represents the second part of the Forge URL for the document, for example, the Forge file ID.

ChangeManager Method

Changes the workflow manager of the document type workflow of the current document object.

Syntax

```
ChangeManager(User As String)
```

Parameters

Name	Description
User	The user name of the new workflow manager

Remarks

For events to occur for this method, the **SDWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

ChangeTodoPerson Method

Changes the to-do person of the document type workflow of the current document object.

Syntax

```
ChangeTodoPerson(User As String, Comment As String)
```

Parameters

Name	Description
User	The user name of the new workflow manager
Comment	The comment to append to the comment log

Remarks

For events to occur for this method, the **SDWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

ChangeWorkflowState Method

Changes the document type workflow state of the current document object.

Syntax

```
ChangeWorkflowState(NewWorkflowState As AS_WF_STATE, User As String,
Comment As String)
```

Parameters

Name	Description
NewWorkflowState	Long integer that represents one or more **AS_WF_STATE** constants
User	The user name to assign as the new to-do person
Comment	The comment to append to the comment log

Remarks

For events to occur for this method, the **SDWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

ClearRedlines Method

Clears the current redlines of the document object. If the document includes a rendition, its redlines are also deleted.

Syntax

```
ClearRedlines()
```

Parameters

This method accepts no parameters.

CreateWaitingProjectCopy Method

Adds the document as a new waiting Project Copy to the waiting list if the document is a master document with concurrent engineering options which allow creation of a waiting Project Copy. All properties in custom property sets are copied from the master document to the new Project Copy. Waiting project copies are created in the workflow state Unchanged.

Syntax

```
CreateWaitingProjectCopy(TargetProject as IASFolder9) As
IASDocument15
```

Parameters

Name	Description
TargetProject	A project folder where the waiting project copy is to reside.

ConfirmMerged Method

Prompts the user to confirm that the selected project copy document has been merged with the master document.

Syntax

```
ConfirmMerged()
```

Parameters

This method accepts no parameters.

ConfirmSuperceded Method

Prompts the user to confirm that the selected project copy document has been superceded by changes made to the master document.

Syntax

```
ConfirmSuperceded()
```

Parameters

This method accepts no parameters.

ContentFromReport Method

Replaces content of the document with the output of the report with the specified name, scope, and template. This only works for reports that generate HTML content. You can use this method in a custom command.

> **Note:**
> We recommend that you do not use this for locked documents as it can result in the new report being lost.

Syntax

```
ContentFromReport(ReportName as String, ReportScope, TemplateName as String)
```

Parameters

Name	Description
ReportName	Internal name of the report definition.
ReportScope	Array of document IDs that represents the scope of the report.
TemplateName	XLST stylesheet to use for the report.

CopyProperties Method

Copies all of the property values of the current document to a specified document except for an optional array of property names.

Syntax

```
CopyProperties(PropertySetName As String, TargetDocument As Document, [ExcludeProperties] As String)
```

Parameters

Name	Description
PropertySetName	Name of the property set to copy.
TargetDocument	Document object to which to copy the properties.
ExcludeProperties	Optional one dimension array of a property names to exclude from copying.

Remarks

The following example copies all properties in the property set **MyPropSet** to the specified document object except for the properties named **Prop1** and **Prop2**.

```
Document.CopyProperties "MyPropSet", Document, Array("Prop1",
"Prop2")
```

CreateHybridPart Method

Creates a hybrid for the current document with the specified name. After creation, content for the new hybrid part can be loaded with the LoadFromFile or LoadFromTemplate methods.

Syntax

```
CreateHybridPart(PartName as String) As IASDocument11
```

Parameters

Name	Description
PartName	The name with which to create the new hybrid part.

Return Value

A new **Document** object.

Remarks

Use this method to create a new hybrid part before loading file content with the LoadFromFile or LoadFromTemplate methods. Consider using the ImportHybridPart method instead.

For events to occur for this method, the **HybridEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

Delete Method

Deletes the current document.

Syntax

```
Delete()
```

Parameters

This method accepts no parameters.

Remarks

The confirmation dialogs that are normally shown during document deletion can be suppressed as described in Confirmation property.

DeleteHotspots method

Deletes the type of hotspots that you specify.

Syntax

```
DeleteHotspots(AS_HOTSPOTS_TYPE)
```

Parameters

Name	Description
AS_ HOTSPOTS_ TYPE	An enumeration that determines whether to delete manual or automatic hotspots. You can use `AS_HOTSPOTS_MANUAL` Or `AS_HOTSPOTS_AUTOMATIC`

DeleteHybridPart Method

Deletes the hybrid part of the current document with the specified name.

Syntax

```
DeleteHybridPart(PartName as String)
```

Parameters

Name	Description
PartName	The name of the hybrid part to delete.

Remarks

For events to occur for this method, the **HybridEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

DeleteRendition Method

Deletes the content of the current rendition revision for the current document.

Syntax

```
Sub DeleteRendition()
```

Parameters

This method accepts no parameters.

Remarks

This method does not actually delete the rendition file but marks it as deleted. The **Document.Rendition** property will then return the path of the orphaned file.

ExecTransition Method

Executes the specified workflow definition transition for the current document object.

Syntax

```
ExecTransition(WorkflowTransition As IASWorkflowTransition, [Comments
As String], [TodoPersons As String])
```

Parameters

Name	Description
WorkflowTransition	An object that represents the workflow definition transition to execute
Comments	Optional comments to append to the comment log
TodoPersons	Optional user names to assign as the new to-do persons

Remarks

- For events to occur for this method, the **CWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).
- Wizard pages are not shown when this method is invoked. Property value assignments should be performed by script instead.
- If the **Release project copies of documents as master revisions** option of a workflow transition is enabled and the transition changes the status of the document to **Released**, executing the transition on a project copy with this method will create a new revision of the project copy as expected but it will not release it as a new master document revision.

ExportRendition Method

Exports the content of the current rendition revision for the current document to a specified file.

Syntax

```
Sub ExportRendition (File As String)
```

Parameters

Name	Description
File	Path and filename outside the vault where to export the rendition content of the current document.

ExtractTags Method

Gets text strings from the full-text index of the current document that match the specified regular expression.

Syntax

`ExtractTags(`*`RegExp`*`)`

Parameters

Name	Description
RegExp	A regular expression that matches the text strings to be found in the full-text index of vault documents. For help calculating a regular expression, see Regular Expression Language - Quick Reference in the MSDN Library.

Return Value

An array of text tokens that match the specified regular expression.

Remarks

This method is intended for finding object tag IDs for use with the Meridian Asset Management Module. This method is equivalent to the **Regular Expression** option described in the *Meridian Enterprise Configuration Guide* that is used with the text tags functionality described in *View Drawing Tags* in the *Meridian Enterprise User's Guide* .

GetDocumentRevisions

Returns an array of revision IDs.

Syntax

```
GetDocumentRevisions() As Array
```

Parameters

This method accepts no parameters.

Example

```
Dim prt
```

```
prt= Document.GetDocumentRevisions()
If IsArray(prt) Then
    Document.Log "array prt"
    Dim i
    For Each i in prt
        Document.Log i
    Next
Else
    Document.Log "not array"
End If
```

GetExpectedTransitionResult Method

Returns the expected result of executing the specified workflow definition transition for the current document object.

Syntax

`GetExpectedTransitionResult(`*`Transition`* `As IASWorkflowTransition) As AS_WORKFLOW_TRANS_RES`

Parameters

Name	Description
Transition	An object that represents the workflow definition transition to execute

Return Value

Returns a long integer that is equivalent to one or more of the **AS_WORKFLOW_TRANS_RES** constants that represents the expected results. Use this method to determine the current consensus status of parallel workflows.

GetHybridPart Method

Gets a document object for the hybrid part with the specified name.

Syntax

```
GetHybridPart(PartName as String) As IASDocument6
```

Parameters

Name	Description
PartName	The name of the hybrid part for which to get a document object.

Return Value

A **Document** object.

GetLog Method

Gets the workflow comment log for the current document object.

Syntax

```
GetLog() As String
```

Parameters

This method accepts no parameters.

Return Value

A single string containing the entire **Comment** property value.

Remarks

This method can be useful to append the workflow history of one document to another that is being replaced or superceded.

Example

```
Sub DocProjectCopyEvent_AfterReleaseToMaster(Batch, MasterDoc,
ProjectCopyChanged)
    MasterDoc.Log Document.GetLog()
```

```
    MasterDoc.ApplyPropertyValues
End Sub
```

GetNavigationViewLevelPath Method

Gets the path of the current document object within a specified navigation view.

Syntax

```
Function GetNavigationViewLevelPath(ViewName)
```

Parameters

Name	Description
ViewName	Name of the view to get the path from.

Return Value

Array containing the names of each level of the path.

GetReferences Method

Gets the references of the specified type for the current document object.

Syntax

```
GetReferences(RefTypeName As String, [InComing As Boolean = False])
As IASReferences
```

Parameters

Name	Description
RefTypeName	The name of the reference type to retrieve.
InComing	If **True**, returns only incoming references. If **False**, returns only outgoing references.

Return Value

A collection of references as described in References Object.

HasHotspots method

Returns a boolean indicating whether automatic or manual hotspots exist on the document.

Syntax

```
HasHotspots(AS_HOTSPOTS_TYPE)
```

Parameters

Name	Description
AS_HOTSPOTS_TYPE	An enumeration that determines whether to check for manual or automatic hotspots. You can use `AS_HOTSPOTS_MANUAL` or `AS_HOTSPOTS_AUTOMATIC`

HybridMainDocument Method

Returns the main document object for the current hybrid part document.

Syntax

```
HybridMainDocument As IASDocument6
```

Return Value

The main document object for the current hybrid part document.

Remarks

Only returns an object if the current document is a part of a hybrid document. Otherwise, it returns Nothing.

ImportHybridPart Method

Creates a new hybrid part for the current document and loads content from the specified file path.

Syntax

```
ImportHybridPart(FilePath as String) As IASDocument11
```

Parameters

Name	Description
FilePath	The path of a file from which to import document content for the new hybrid part.

Return Value

A **Document** object.

Remarks

This method is equivalent to the combination of the **CreateHybridPart** and **LoadFromFile** methods.

For events to occur for this method, the **HybridEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

LinkToMaster Method

Links the selected document to a specified master document from which it was not originally copied.

Syntax

```
LinkToMaster(MasterDocument As Object, [LockMaster As Boolean=False])
```

Parameters

Name	Description
MasterDocument	A document object to which to link the selected document.
LockMaster	If **True**, locks the master document after the link has been made. Optional.

Remarks

Ideally, this method should not be used under the same conditions as the **Link to Master** command described in *Link To a Master Document* in the *Meridian Enterprise User's Guide*.

LoadFromFile Method

Loads content for the current document from the specified file.

Syntax

```
LoadFromFile(FileName as String)
```

Parameters

Name	Description
FileName	If specified, the file to load from the specified location outside the vault. If absent, the file is loaded from the Local Workspace with its current name.

Return Value

The name of the file that was loaded.

Remarks

Use this method to load content for a newly created hybrid part.

LoadFromTemplate Method

Loads content for the current document from the specified document type template.

Syntax

```
LoadFromTemplate(TemplateName as String)
```

Parameters

Name	Description
TemplateName	If specified, the file is saved in the specified location outside the vault. If absent, the file is saved in the Local Workspace with its current name.

Remarks

Use this method to load content for a newly created hybrid part.

LockLWS Method

Locks the current document object in the user's local workspace.

Syntax

```
LockLWS()
```

Parameters

This method accepts no parameters.

Return Value

The path of the file that was locked.

LockMasterDocument Method

Locks the master document from which the current project copy was made.

Syntax

```
LockMasterDocument()
```

Parameters

This method accepts no parameters.

Log Method

Appends a line of text to the **Comments** property of the current document object.

Syntax

```
Log(Line As String)
```

Parameters

Name	Description
Line	A line of text to add to the document's comment log

Migrate Method

Migrates the current document object from an active document type workflow state to the specified workflow definition state.

Syntax

```
Migrate(Workflow As String, [WorkFlowState], [Comments])
```

Parameters

Name	Description
Workflow	The name of the workflow definition to which to migrate the document.
WorkFlowState	Optional workflow definition state name to which to migrate the document.
Comments	Optional text to append to the document's comment log.

MoveTo Method

Moves the current document to the specified folder.

Syntax

```
MoveTo(TargetFolder As IASFolder3, [Options As Long = 0])
```

Parameters

Name	Description
TargetFolder	An object that represents the folder to which to move the current document object.
Options	Optional long integer that represents one or more **AS_MOVE_OPTIONS** constants.

Remarks

If this method is called within the **AfterExecuteTransition** event, the text **Copy of** is prepended to the filename if the document already resides in the target folder. This method does not support Field-Path definition levels.

When this method is called, any property values set by code are reset to their original values.

If this method is used in event procedures such as **DocGenericEvent_AfterNewDocument**, an unhandled exception **Cannot delete a relation. The object is already being edited by another session** can occur. This can happen if the folder to which the document is being moved is locked, such as by another new document wizard in use by another user at the same time. This is likely to happen if event procedures are customized to provide automatic document storage together with the Meridian Advanced Project Workflow Module so that it works similar to a Field-Path definition.

For the wizard events to occur for this method, the **MoveEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

> **Note:**
> To use this method to move documents to folders that users are not granted **Create Document** privilege, grant them the **Create Document from Script** privilege for the destination folder. The privilege may only be granted for custom folder types , not normal folders.

ReassignManagers Method

Reassigns the workflow definition managers of the current document object.

Syntax

```
ReassignManagers(Managers, [Comments])
```

Parameters

Name	Description
Managers	The user names of the new workflow managers.
Comments	Optional comments to add to the document's **Comments** property.

Remarks

For events to occur for this method, the **CWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

ReassignTodoPersons Method

Reassigns the workflow definition to-do persons of the current document object.

Syntax

```
ReassignTodoPersons(Persons, [Comments])
```

Parameters

Name	Description
Persons	The user names of the new to-do persons.
Comments	Optional comments to add to the document's **Comments** property.

Remarks

For events to occur for this method, the **CWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

ReleaseChange Method

Releases the working copy or Quick Change of the current document object.

Syntax

```
ReleaseChange()
```

Parameters

This method accepts no parameters.

Remarks

For events to occur for this method, the **QuickChangeEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

Reroute Method

Reroutes the current document object to a specified workflow definition state.

Syntax

```
Reroute(WorkFlowState, [Comments], [ToDoPersons])
```

Parameters

Name	Description
WorkFlowState	The name of the state to which to reroute the document.
Comments	Optional comments to add to the document's **Comments** property.
ToDoPersons	Optional user names of the new to-do persons.

Remarks

For events to occur for this method, the **CWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

ResetPropertyValues Method

Resets the properties of the current document object.

Syntax

```
ResetPropertyValues()
```

Parameters

This method accepts no parameters.

RevokeChange Method

Revokes the working copy or Quick Change of the current document object.

Syntax

```
RevokeChange()
```

Parameters

This method accepts no parameters.

Remarks

For events to occur for this method, the **QuickChangeEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

RevokeWorkflow Method

Revokes the active workflow definition of the current document object.

Syntax

```
RevokeWorkflow()
```

Parameters

This method accepts no parameters.

Remarks

For events to occur for this method, the **SDWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

SaveToFile Method

Saves the current document content to an external file or Local Workspace file.

Syntax

```
SaveToFile([FileName as String])
```

Parameters

Name	Description
FileName	If specified, the file is saved in the specified location outside the vault. If absent, the file is saved in the Local Workspace with its current name.

Return Value

The path of the file that was saved.

Remarks

Use this method to temporarily save the content of a non-hybrid part document before creating a new hybrid part and then loading the file created by this method as the new part's content.

SendToPortal Method

Sends the current document to the Meridian Portal project to which the document's parent folder is linked.

Syntax

```
SendToPortal(Recipient As String, IncludeContent As AS_INCLUDE_
CONTENT_OPTIONS, Description As String, Options As AS_PORTAL_OPTIONS)
```

Parameters

Name	Description
Recipient	A string containing name of recipient in Portal project.
IncludeContent	A long integer that represents one or more AS_INCLUDE_CONTENT_OPTIONS constants.
Description	A string containing the package description.
Options	A long integer that represents one or more AS_PORTAL_OPTIONS constants.

Remarks

To learn more about sending documents to Meridian Portal and the parameters in the table above, see *Send Documents To Meridian Portal* in the *Meridian Enterprise User's Guide*.

SetModified Method

Changes the internal **Modified** property of a document.

Syntax

```
SetModified(ModifiedDate As Date)
```

Parameters

Name	Description
ModifiedDate	The date to set for the **Modified** property.

Remarks

This method can be useful in **DocProjectCopy_*ReleaseToMaster** event procedures when a project copy is released as a new master document but the document has no content, such as asset tags. Setting the **Modified** property of a project copy causes Meridian to consider the document as changed and the master document is replaced.

StartChange Method

Initiates a working copy or Quick Change of the current document object.

Syntax

```
StartChange()
```

Parameters

This method accepts no parameters.

Remarks

For events to occur for this method, the **QuickChangeEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

StartWorkflow Method

Initiates a workflow definition of the current document object.

Syntax

```
StartWorkflow(Workflow As String, [Transition], [Comments],
[TodoPersons], [Managers])
```

Parameters

Name	Description
Workflow	The name of the workflow definition to initiate.
Transition	Optional name of a transition to execute.
Comments	Optional comments to add to the document's **Comments** property.
ToDoPersons	Optional user names of the to-do persons.
Managers	Optional user names of the workflow managers.

Remarks

For events to occur for this method, the **CWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

Subscribe Method

Registers the specified user to receive notifications of events for the current document.

Syntax

```
Subscribe(User As String, Notification As String)
```

Parameters

Name	Description
User	User (To-Do list) name or array of names.
Notification	Internal name or array of internal names of event notification definitions to send to the specified users.

Unsubscribe Method

Unregisters the specified user from receiving notifications of events for the current document.

Syntax

```
Unsubscribe(User As String, Notification As String)
```

Parameters

Name	Description
User	User (To-Do list) name or array of names.
Notification	Internal name or array of internal names of event notification definitions to unregister for the specified users.

UnlockLWS Method

Unlocks the current document object from the user's local workspace.

Syntax

```
UnlockLWS()
```

Parameters

This method accepts no parameters.

Return Value

The path of the file that was unlocked.

UnLockMasterDocument Method

Unlocks the master document from which the current project copy was made.

Syntax

```
UnLockMasterDocument()
```

Parameters

This method accepts no parameters.

UpdateRendition Method

Requests an external process to update the rendition content for the current document.

Syntax

```
Sub UpdateRendition()
```

Remarks

An interface for this method to specify the external process has not yet been implemented.

DocumentType Object

The **DocumentType** object represents a document type definition.

DocumentType Object Properties

The **DocumentType** object provides the following properties.

ConfigKeywords Property

Gets the keywords configured for a document type.

Syntax

```
ConfigKeywords As String
```

Remarks

Assign keywords to document types to make it convenient to test for them in more ways than just by name. This can be useful for associating multiple document types together, for example, to authorize access by users or groups. For information about setting keywords in document types, see *Configure Document Type General Options* in the *Meridian Enterprise Configuration Guide*.

Example

```
If InStr(document.DocumentType.ConfigKeywords, "MyKeyword",1) > 0
Then
...
End If
```

DisplayName Property

The **Display Name** property of the document type. Read-only.

Syntax

```
DisplayName As String
```

InternalName Property

The internal name of the document type.

Syntax

```
InternalName As String
```

Sequence Property

An object that represents the specified document type sequence.

Syntax

```
Sequence(Name) As IASSequence
```

> **Important!**
> Using the vault name as a sequence name can cause errors and possibly crash the Accruent EDM Server service.

Remarks

For information about the **Sequence** object, see Sequence Object. For information about using sequences in file names, see *File Name Calculation* in the *Meridian Enterprise Configuration Guide*.

ExportPackage Object

The **ExportPackage** object is a property of the **ExportPackages** object and represents the current export package.

ExportPackage Object Properties

The **ExportPackage** object provides the following properties, some of which are read-only.

ContentIncluded Property

Gets or sets a value that indicates the types of document content contained in the current export package.

Syntax

```
ContentIncluded As Long
```

Return Values

One of the **AS_EXPORTPACKAGE_CONTENTINCLUDED_VALUES** constants that are available in the VBScript editor **Object Browser**.

Remarks

The package must be in the **Open** status to set the value.

Description Property

Gets or sets the description of the current package.

Syntax

```
Description As String
```

Remarks

The package must be in the **Open** status to set the value.

Destination Property

Gets or sets name of the publishing job assigned to the current export package.

Syntax

```
Destination As String
```

Remarks

The package must be in the **Open** status to set the value.

ID Property

Returns the ID of the current object. The value is unique within the current vault. For the ID value that is unique among all vaults, retrieve the **GlobalID** property value, if available. Read-only.

Syntax

```
ID As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager*

Meridian\CurrentVersion\Server\UserDatabase article in the *Meridian Enterprise Administrator's Guide*.

PackageURL Property

Gets the URL of the Meridian Explorer page to show the current export package. Read-only.

Syntax

```
PackageURL As String
```

ProjectID Property

Gets the Meridian Portal project ID (GUID) of the current package. Only available after the Meridian Enterprise project has been linked to a Meridian Portal project. Read only.

Syntax

```
ProjectID As Long
```

Remarks

The ID is stored in the Meridian Enterprise folder property **ProjectFolder PropertySet.ProjectID**. The ID can be set by the SetProjectID method.

ProjectName Property

Gets the Meridian Enterprise project name of the current package. Read only.

Syntax

```
ProjectName As String
```

Recipient Property

Gets or sets the recipient of the current export package.

Syntax

```
Recipient As String
```

Remarks

The package must be in the **Open** status to set the value.

Status Property

Gets the status of the current export package. Read-only.

Syntax

```
Status As Long
```

Return Values

One of the **AS_EXPORTPACKAGE_STATUS_VALUES** constants that are available in the VBScript editor **Object Browser**.

ExportPackage Object Methods

The **ExportPackage** object methods affect the current export package.

Abort Method

Aborts sending the current export package.

Syntax

```
Abort()
```

Remarks

The package must be in the **Failed** or **Pending** status.

AddDocument Method

Adds the specified document to the current export package.

Syntax

```
AddDocument(DocGlobalID As String)
```

Parameters

Name	Description
DocGlobalID	The **GlobalID** value of the document to add.

Remarks

Only supported for export packages in the **Open** status.

Close Method

Changes the status of the current export package to **Closed**.

Syntax

```
Close()
```

Remarks

The package must be in the **Sent**, **Failed**, or **Open** status.

GetDocuments Method

Gets the documents contained in the current package.

Syntax

```
GetDocuments() As Array
```

Return Value

Returns an array of **GlobalID** values for the documents.

Remarks

When the current package is an import package, returns an array only if the **Status** property of the package is **AS_IPS_Imported**.

IsDocumentInPackage Method

Gets a value that indicates whether the specified document is contained in the current export package.

Syntax

```
IsDocumentInPackage(DocGlobalID As String) As Boolean
```

Parameters

Name	Description
DocGlobalID	The **GlobalID** value of the document to check.

Return Value

True if the export package contains the specified document, otherwise **False**.

IsDocumentOutdatedInPackage Method

Gets a value that indicates whether modification time of the specified document is later than the time when it was exported to the package.

Syntax

```
IsDocumentOutdatedInPackage(DocGlobalID As String) As Boolean
```

Parameters

Name	Description
DocGlobalID	The **GlobalID** value of the document to check.

Return Value

True if the document in the export package is outdated, otherwise False.

RemoveDocument Method

Removes the specified document from the current export package.

Syntax

```
RemoveDocument(DocGlobalID As String)
```

Parameters

Name	Description
DocGlobalID	The **GlobalID** value of the document to remove.

Remarks

Only supported for export packages in the **Open** status.

Reopen Method

Changes the status of the current export package from the **Closed** state to the **Open** state.

Syntax

```
Reopen()
```

Remarks

The package must be in the **Sent** or **Closed** status.

Send Method

Sends the current package to the destination specified in the publishing job.

Syntax

```
Send()
```

Remarks

The package must be in the **Open** status.

SetProjectID Method

Sets the Meridian Portal project ID (GUID) of the current package.

Syntax

```
SetProjectID (ProjectID as String)
```

Remarks

The ID can be retrieved by the ProjectID property.

Update Method

Saves the properties of the current package.

Syntax

```
Update()
```

Remarks

Invoke this method after setting all of the package property values and before adding documents and sending the package.

ExportPackages Object

The **ExportPackages** object exposes the export package functionality of the vault. The **ExportPackages** object is available to the **Vault** object.

ExportPackages Object Methods

The **ExportPackages** object methods affect the collection of all **ExportPackage** objects related to the current vault.

CreateNewPackage Method

Creates a new export package for documents from the current vault.

Syntax

```
CreateNewPackage(Name As String) As IASExportPackage
```

Parameters

Name	Description
Name	The name of the new export package

Return Value

An **ExportPackage** object.

FindDocumentPackages Method

Gets the export packages that contain the document with a given **GlobalID** value.

Syntax

```
FindDocumentPackages(DocGlobalID As String) As Array
```

Parameters

Name	Description
DocGlobalID	The **GlobalID** value of the document for which to find the related export packages.

Return Value

An array of **ExportPackage** objects.

FindPackage Method

Gets an export package with a given name.

Syntax

```
FindPackage(NameOrID As String) As IASExportPackage
```

Parameters

Name	Description
NameOrID	Name or ID of the export package to get.

Return Value

An **ExportPackage** object.

Folder Object

The **Folder** object represents the selected folder or the parent folder of the selected **Document** object and is available for event procedures that apply to folders.

Folder Object Properties

The **Folder** object provides the following properties, most of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

The **AMProjectWorkflowPropertySet** property set is assigned to a folder type when a workflow definition is assigned to the folder type. If the workflow definition is unassigned later, the property set will remain assigned to the folder type and cannot be removed. This is necessary to support vault history and is similar to property sets assigned to document types. Also, if the folder type is assigned to multiple levels of a Field-Path definition, the values returned to documents in the structure will be from the immediately previous level. Because the value can be inconsistent, routed properties should only be used in vault customization with great care.

AutoDocumentType Property

Gets or sets the name of the document type to apply to documents that are automatically created in the current folder.

Syntax

```
AutoDocumentType As String
```

Remarks

The documents may be created because the folder is a shared workspace and the documents were created in the linked folder that resides outside the vault. The documents are then imported into the vault when the shared workspace is synchronized with the vault. For more information about configuring shared workspaces, see *Configure Shared Workspaces* in the *Meridian Enterprise Configuration Guide* .

Created Property

The date and time when the folder was created. Read-only.

Syntax

```
Created As Date
```

CreatedBy Property

An object that represents the user who created the folder. Read-only.

Syntax

```
CreatedBy As IASUser
```

HasDocuments Property

True if the folder or its sub folder contains documents. Read-only.

Syntax

```
HasDocuments As Boolean
```

ID Property

Returns the ID of the current object. The value is unique within the current vault. For the ID value that is unique among all vaults, retrieve the **GlobalID** property value, if available. Read-only.

Syntax

```
ID As String
```

IsProject Property

True if the folder is associated with a project definition. Read-only.

Syntax

```
IsProject As Boolean
```

IsUniqueValue Property

True if the specified value for the specified folder property is unique within the current vault. Read-only.

Syntax

```
IsUniqueValue(PropertyName, Value) As Boolean
```

LinkedProjectID Property

ProjectFolderPropertySet.ProjectID value for current project folder. Read-only.

Syntax

```
LinkedProjectID As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

ParentFolder Property

An object that represents the parent folder of the folder. Read-only.

Syntax

```
ParentFolder As IASFolder3
```

ParentProject Property

An object that represents the parent project folder of the current project object. Read-only.

Syntax

```
ParentProject As IASFolder3
```

Path Property

Relative path of the folder from the root of the vault. Read-only.

Syntax

```
Path As String
```

ProjectTypeDisplayName Property

The name of the project type as seen by users. Read-only.

Syntax

```
ProjectTypeDisplayName As String
```

ProjectTypeName Property

The internal name of the current project type object. Read-only.

Syntax

```
ProjectTypeName As String
```

Property Property

The value of the specified (String) folder property.

Syntax

```
Property(Name) As Variant
```

Roles Property

An object that represents the security roles that are applied to the current folder object.

For information about the **Roles** object, see Roles Object. For information about configuring security roles, see *Security Roles* in the *Meridian Enterprise Configuration Guide*. For information about assigning security roles to folders, see *Assign Security Roles To a Folder* in the *Meridian Enterprise User's Guide*.

Sequence Property

An object that represents the specified folder sequence.

> **Important!**
> Using the vault name as a sequence name can cause errors and possibly crash the Meridian EDM Server service.

For information about the **Sequence** object, see Sequence Object. For information about using sequences in file names, see *File Name Calculation* in the *Meridian Enterprise Configuration Guide*.

Syntax

```
Sequence(Name) As IASSequence
```

ShareID Property

A server ShareID to get a transaction object from another process. Read-only.

Syntax

```
ShareID As String
```

TypeDisplayName Property

The **Display Name** value of the folder type. Read-only.

Syntax

```
TypeDisplayName As String
```

TypeName Property

The **Name** value of the folder type. Read-only.

Syntax

```
TypeName As String
```

WorkflowManagers Property

The names of the managers of the current workflow object. This property is a string when there is a single user and a string array when multiple users are assigned. You can test with the function **IsArray(Folder.WorkflowManagers)** whether there are one or multiple users. Read-only.

Syntax

```
WorkflowManagers As String
```

WorkFlowState Property

Long integer that represents one or more **AS_WF_STATE** constants. Read-only.

Syntax

```
WorkFlowState As AS_WF_STATE
```

WorkflowStatus Property

A long integer that represents one or more **AS_PWF_STATUS** constants. Read-only.

Syntax

```
WorkflowStatus As AS_PWF_STATUS
```

Folder Object Methods

The **Folder** object methods affect the current folder object.

ApplyPropertyValues Method

Saves the current property values to the current folder object.

Syntax

```
ApplyPropertyValues()
```

Parameters

This method accepts no parameters.

ChangeFolderType Method

Changes the folder type of the current folder object.

Syntax

```
ChangeFolderType(NewFoldertype As String)
```

Parameters

Name	Description
NewFolderType	The name of the new folder type.

Remarks

For events to occur for this method, the **ChangeFolderTypeEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

ConvertToProject Method

Converts the current folder object to a specified project folder.

Syntax

```
ConvertToProject (ProjectDefinitionName As String)
```

Parameters

Name	Description
ProjectDefinitionName	The name of the project definition.

CopyProperties Method

Copies all of the property values of the current folder to a specified folder except for an optional array of property names.

Syntax

```
CopyProperties(PropertySetName As String, TargetFolder As Folder,
[ExcludeProperties] As String)
```

Parameters

Name	Description
PropertySetName	Name of the property set to copy.
TargetFolder	Folder object to which to copy the properties.
ExcludeProperties	Optional one dimension array of a property names to exclude from copying.

Remarks

The following example copies all properties in the property set **MyPropSet** to the specified folder object except for the properties named **Prop1** and **Prop2**.

```
Folder.CopyProperties "MyPropSet", Folder, Array("Prop1", "Prop2")
```

Delete Method

Deletes the folder.

Syntax

```
Delete()
```

ExecTransition Method

Executes the specified project workflow transition for the current folder object.

Syntax

```
ExecTransition(WorkflowTransition As IASWorkflowTransition, [Comments
As String], [Managers As String])
```

Parameters

Name	Description
WorkflowTransition	An object that represents the project workflow transition to run.
Comments	Optional comments to append to the comment log.
Managers	Optional user names to assign as the new workflow managers.

Remarks

For events to occur for this method, the **PrjWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**(default).

FindDocuments Method

Used to search the documents in a folder.

Syntax

```
FindDocuments (Wildcard As String, [DocumentTypes],
[PropertyFilters], [LogicalOR As Boolean = False], [SearchScope As
Integer = AS_SEARCHSCOPE_CURRENTFOLDER) As IASDocuments
```

Parameters

Name	Description
SearchScope	There are two constants that can be used for this parameter: AS_SEARCHSCOPE_ CURRENTFOLDER or AS_SEARCHSCOPE_RECURSIVE. • **AS_SEARCHSCOPE_CURRENTFOLDER** is used to search only the current folder • **AS_SEARCHSCOPE_RECURSIVE** is used to search the current folder and any subfolders

GetExpectedTransitionResult Method

Returns the expected result of executing the specified project workflow transition for the current folder object.

Syntax

```
GetExpectedTransitionResult(Transition As IASWorkflowTransition) As
AS_WORKFLOW_TRANS_RES
```

Parameters

Name	Description
Transition	An object that represents the project workflow transition to run.

Remarks

Returns a long integer that is equivalent to one or more of the **AS_WORKFLOW_TRANS_RES** constants that represents the expected results. Use this method to determine the current consensus status of parallel workflows.

GetSubFolder Method

Returns a folder object that represents a subfolder of the current folder object.

Syntax

```
GetSubFolder(Name As String, [TypeName As String], [Options As Long =
0]) As IASFolder6
```

Parameters

Name	Description
Name	The name of the subfolder to return.
TypeName	Folder type of the subfolder to return. To return a standard folder, specify an empty string.
Options	Optional long integer that represents one or more **AS_NEWFOLDER_OPTIONS** constants.

Return Value

A **Folder** object.

Remarks

Pass the **AS_NFO_CREATE_IFNOTEXIST** constant to this method to create a new subfolder equivalent to the **NewFolder** method.

For the wizard events to occur for this method, the **NewWizardEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

GetSubFolderNames Method

Gets the names of the subfolders of the current folder object.

Syntax

```
GetSubFolderNames([FolderType As String])
```

Parameters

Name	Description
FolderType	Optional folder type of the subfolders to return.

Return Value

An array of subfolder names.

GetUsersWithRole Method

Gets all users with a specified security role for the current folder.

Syntax

```
GetUsersWithRole (Role As String)
```

Parameters

Name	Description
Role	The name of the role for which to get users.

Return Value

An array of strings containing the user names.

Interlock Method

Returns the result of evaluating the specified interlock.

Syntax

```
Interlock(InterlockName As String, [InterlockLocationRLOCK_LOCATION =
AS_WFIL_PARENT_PROJECT], [DefaultResult As Boolean = False]) As
Boolean
```

Parameters

Name	Description
InterlockName	The name of the interlock to evaluate.
InterlockLocation	The location of the interlock to evaluate expressed as one of the **AS_ WFINTERLOCK_LOCATION** constants.
DefaultResult	A value to return by default if the interlock is not satisfied.

LinkToPortal Method

Links the Meridian project to a Meridian Portal project

Syntax

```
LinkToPortal()
```

Parameters

This method accepts no parameters.

MoveTo Method

Moves the current folder to the specified folder.

Syntax

```
MoveTo(TargetFolder As IASFolder3, [Options As Long = 0], [Comments])
```

Parameters

Name	Description
TargetFolder	An object that represents the folder to which to move the current document object.
Options	An optional long integer that represents the move options. One or more of the **AS_MOVE_OPTIONS** constants.

NewFolder Method

Creates a new subfolder of the current folder object.

Syntax

```
NewFolder(FolderName As String, TypeName As String, [Options As Long
= 0]) As IASFolder
```

Parameters

Name	Description
FolderName	The name of the new folder.
TypeName	Folder type name for the new folder. To create a standard folder, specify an empty string.
Options	Optional long integer that represents one or more **AS_NEWFOLDER_OPTIONS** constants.

Return Value

A **Folder** object.

Remarks

The **AS_NFO_CREATE_IFNOTEXIST** constant can also be passed to the **GetSubFolder** method to create a new subfolder.

The **AS_NFO_SHOWWIZARD** is not supported in PowerWeb.

ReassignWorkflowManagers Method

Reassigns the workflow definition managers of the current folder object.

Syntax

```
ReassignWorkflowManagers(Managers, [Comments])
```

Parameters

Name	Description
Managers	The user names of the new workflow managers.
Comments	Optional comments to add to the document's **Comments** property.

Remarks

For events to occur for this method, the **PrjWFEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

Reroute Method

Reroutes the current folder object to a specified project workflow state.

Syntax

```
Reroute(WorkFlowState, [Comments], [Managers])
```

Parameters

Name	Description
WorkFlowState	The name of the state to which to reroute the folder.
Comments	Optional comments to add to the folder's **Comments** property.
Managers	Optional user names of the new workflow managers.

ResetPropertyValues Method

Resets the properties of the current folder object.

Syntax

```
ResetPropertyValues()
```

Parameters

This method accepts no parameters.

UnlinkFromPortal Method

Removes the link from the Meridian project to Meridian Portal project.

Syntax

```
UnlinkFromPortal()
```

Parameters

This method accepts no parameters.

ImportPackage Object

The **ImportPackage** object represents the current import package.

ImportPackage Object Properties

The **ImportPackage** object provides the following properties, all of which are read-only. They are described below.

Description Property

Gets or sets the description of the current package.

Syntax

```
Description As String
```

Remarks

The package must be in the **Open** status to set the value.

ID Property

Returns the ID of the current object. The value is unique within the current vault. For the ID value that is unique among all vaults, retrieve the **GlobalID** property value, if available. Read-only.

Syntax

```
ID As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

ImportProfile Property

Gets the **ImportProfile** object that is assigned to the current **ImportPackage** object.

Syntax

```
ImportProfile As Object
```

ImportPackage Object Methods

The **ImportPackage** object methods affect the current import package. They are described below.

GetDocuments Method

Gets the documents contained in the current package.

Syntax

```
GetDocuments() As Array
```

Return Value

Returns an array of **GlobalID** values for the documents.

Remarks

When the current package is an import package, returns an array only if the **Status** property of the package is **AS_IPS_Imported**.

ImportProfile Object

The **ImportProfile** object represents an existing package import profile that is defined in Meridian Enterprise Server.

ImportProfile Object Properties

The **ImportProfile** object provides the following properties, all of which are read-only. They are described below.

Destination Property

Gets or sets the name of the destination vault that is assigned to the current **ImportProfile**.

Syntax

```
Destination As String
```

ID Property

Returns the ID of the current object. The value is unique within the current vault. For the ID value that is unique among all vaults, retrieve the **GlobalID** property value, if available. Read-only.

Syntax

```
ID As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

MailMessage Object

The **MailMessage** object represents an email message. The **MailMessage** object is available to the **Briefcase** object.

MailMessage Object Properties

The **MailMessage** object provides the following properties, most of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

Attachments Property

A collection of attachments for the current message. Read-only.

Syntax

```
Attachments As IASAttachments
```

NoteText Property

The body text of the current message.

Syntax

```
NoteText As String
```

Originator Property

An object that represents the originator of the current message. Read-only.

Syntax

```
Originator As IASMailRecipient
```

Recipients Property

A collection of recipients of the current message. Read-only.

Syntax

```
Recipients As IASMailRecipients
```

Subject Property

The subject text of the current message.

Syntax

```
Subject As String
```

MailMessage Object Methods

The **MailMessage** object methods affect the current **MailMessage** object. They are described below.

Clean Method

Removes all properties from the current **MailMessage** object.

Syntax

```
Clean()
```

Parameters

This method accepts no parameters.

Send Method

Sends the current **MailMessage** object.

Syntax

```
Send(Flags As Long)
```

Parameters

Name	Description
Flags	A combination of one or more **AS_MAPIMSG_SEND_FLAGS** constants. Read-only.

MailRecipient Object

The **MailRecipient** object represents a recipients of an email message. The **MailRecipient** object is available to the current **MailRecipients** collection.

MailRecipient Object Properties

The **MailRecipient** object provides the following properties.

Address Property

The email address of the recipient.

Syntax

```
Address As String
```

Remarks

The **Name** property should be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with

Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

Type Property

The type of recipient as one of the **AS_MAPIMSG_RECIP_TYPE** constants.

Syntax

```
Type As AS_MAPIMSG_RECIP_TYPE
```

MailRecipients Object

The **MailRecipients** object represents a collection of recipients of an email message. The **MailRecipients** object is available to the current **Briefcase** object.

MailRecipients Object Properties

The **MailRecipients** object provides the following properties, all of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

Count Property

The total number of message recipients. Read-only.

Syntax

```
Count As Long
```

Recipient Property

An object that represents the individual recipient specified by **Index**.

Syntax

```
Recipient(Index) As IASMailRecipient
```

MailRecipients Object Methods

The **MailRecipients** object methods affect the current **MailRecipients** object. They are described below.

Add Method

Adds a recipient to the current **MailRecipients** collection.

Syntax

```
Add(Address As String, [Type As AS_MAPIMSG_RECIP_TYPE = AS_MMRT_TO],
[Name As String]) As IASMailRecipient
```

Parameters

Name	Description
Address	The email address of the recipient to add.
Type	The type of recipient as one of the **AS_MAPIMSG_RECIP_TYPE** constants.
Name	The name of the recipient to add.

Return Value

A **MailRecipient** object.

Remarks

The **Name** parameter should be used to specify the recipient's email address and an empty string passed to the **Address** parameter. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, **Name** parameter may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Examples

```
oMessage.Recipients.Add "" ,AS_MMRT_TO, "<" + Document.EmailAddress + ">"

oMessage.Recipients.Add "" ,AS_MMRT_TO, "User Full Name" + "<" + Document.EmailAddress + ">"

oMessage.Recipients.Add "" ,AS_MMRT_TO, "User Last Name"
```

Remove Method

Removes a recipient from the current **MailRecipients** object.

Syntax

`Remove(`*`Index`*`)`

Parameters

Name	Description
Index	The recipient in the **MailRecipients** collection specified by **Index**. Read-only.

RemoveAll Method

Removes all recipients from the current **MailRecipients** collection.

Syntax

```
RemoveAll()
```

Parameters

This method accepts no parameters.

MasterDocument Object

A **Document** object that represents the master document from which the current project copy document was copied. The methods of the **MasterDocument** object are described below.

MasterDocument Object Methods

The **MasterDocument** object has the following methods.

TransferToNext Method

Activates the next project copy in the waiting list of a master document. If the concurrent engineering rule property is set to `AS_CER_SERIAL_WITH_RELEASE`, the property `Waiting PC Priority` is set to **0** for the activated project copy.

Syntax

`TransferToNext`

Example

In this example, the vault is configured to use a property called `AutoTransfer`, which is set by a user on the project folder. This is used to determine whether to execute the `TransferToNext` method.

```
Sub DocProjectCopyEvent_AfterReleaseToMaster(Batch, MasterDoc, ProjectCopyChanged)

    MasterDoc.ReleasedBy = Vault.User

    MasterDoc.ReleasedAt = Year (Today) & "-" & Right ("0" & Month (Today), 2) & "-" & Right ("0" &
Day (Today), 2)

    MasterDoc.ReleasedFromProject = Document.ProjectName

    MasterDoc.ProjectName = ""

    MasterDoc.ApplyPropertyValues


    If Document.ParentProject.Project_AutoTransfer Then

        MasterDoc.TransferToNext

    End If

End Sub
```

MeridianQueue Object

The Publisher queue can be managed using the **MeridianQueue** object methods that are described below.

MeridianQueue Object Methods

The **MeridianQueue** object provides the methods that are described below.

BeginBatch Method

Begins collection of a batch of documents to register in the Publisher queue.

Syntax

```
Function BeginBatch (vaultId, commitSize)
```

Parameters

Name	Description
vaultId	A string that uniquely identifies the vault in which the document resides. Specify **Nothing** or an empty string to work in the same vault where the VBScript runs. The vault identifier has the syntax **\\server\vault**, where **server** is the name of the server computer and **vault** is the internal name of the vault or datastore. The vault identifier may consist of only a vault name, in which case the vault's default server, if set, will be assumed.
commitSize	The number of documents to send to the server as a single transaction. The default is **0**, all documents in the batch are sent as one transaction.

Remarks

This method must be called in the **_Initialize** event procedure of a custom command, similar to the following examples. If this is not done, then the **RegisterDocument** method registers documents in the queue individually.

If any document is registered more than once, the Publisher configuration options described in *Configure a Vault For Publisher* in the *Meridian Enterprise Administrator's Guide* will determine the disposition of the batch.

Example

The following examples demonstrate how to do batch publishing either in a server-side process (better performance and recommended) or a client-side process (if a client context is required).

> **Note:**
> The server-side example relies on the account under which the EDM Server service is running to have access to the Meridian Enterprise Server web service.

```
Sub Publish_Batch_Execute(Batch)
       AddToBatch Batch, Document
End Sub

Sub Publish_Batch_Initialize(Batch)
    BeginBatch(Batch)
End Sub

Sub Publish_Batch_Terminate(Batch)
  CommitBatch(Batch)
  If Err.Number = 0 Then
       WinMsgBox ("Documents have been registered for publishing")
  End If
  If Err.Number <> 0 Then
         WinMsgBox Err.Description
      Err.Clear
  End If

End Sub

Public Sub BeginBatch(Batch)
  Dim QueueObject
  ' Server side batch. For client side batch, pass False instead
  Set QueueObject = AMCreateObject("BCPublisher.MeridianQueue", True)
  Batch.Argument("PublisherBatch") = QueueObject
  QueueObject.BeginBatch
End Sub

Public Sub CommitBatch(Batch)
  On Error Resume Next
  Dim QueueObject
  Set QueueObject = Batch.Argument("PublisherBatch")
  Batch.Argument("PublisherBatch") = Nothing
  If Not (QueueObject Is Nothing) Then
    QueueObject.CommitBatch
    QueueObject.Dispose()
  End If

  If Err.Number <> 0 Then
```

```
    Err.Raise Err.Number, Err.Source, _
      "Failed to publish batch of documents: " + Err.Description
  End If
End Sub

Public Sub AddToBatch(Batch, Document)
  Dim QueueObject
  Set QueueObject = Batch.Argument("PublisherBatch")
  If Not (QueueObject Is Nothing) Then
    Call QueueObject.RegisterDocument("", "1B4423", Document.ID, , ,
, , ,"")
  End If
End Sub
```

CommitBatch Method

Commits as one batch all documents that have been registered in the Publisher queue since the calling of the **BeginBatch** method.

Syntax

```
Function CommitBatch (vaultId)
```

Parameters

Name	Description
vaultId	A string that uniquely identifies the vault in which the document resides. Specify **Nothing** or an empty string to work in the same vault where the VBScript runs. The vault identifier has the syntax **\\server\vault**, where **server** is the name of the server computer and **vault** is the internal name of the vault or datastore. The vault identifier may consist of only a vault name, in which case the vault's default server, if set, will be assumed.

Remarks

This method must be called in the **_Terminate** event procedure of a custom command, similar to the example shown in BeginBatch method. If this is not done, then the **RegisterDocument** method registers documents in the queue individually.

RegisterDocument Method

Registers a Meridian Enterprise document in the Publisher queue.

Syntax

```
Function RegisterDocument(vaultId, jobId, documentId, _
                          revisionId, publishOptions, renderOptions,
_
                          userName, preventDuplicate,
feedbackProperty,
                          customColumns)
```

Parameters

Name	Description
vaultId	A string that uniquely identifies the vault in which the document resides. Specify **Nothing** or an empty string to work in the same vault where the VBScript operates. The vault identifier has the syntax **\\server\vault**, where **server** is the name of the server computer and **vault** is the internal name of the vault or datastore. The vault identifier may consist of only a vault name, in which case the vault's default server, if set, will be assumed.
jobId	A string that uniquely identifies the publishing job name. The document is published to all of the destination systems.
documentId	A string that uniquely identifies the document to publish. This parameter accepts either a document ID, path, or Global ID value. However, because the document's Global ID and path can change, we do not recommend using values other than the document ID except under the direction of Accruent Technical Support.
revisionId	A string that uniquely identifies the revision of the document. The default is **Nothing**.
publishOptions	A string that specifies the options for the system links. The default is **Nothing**. For information about the options that can be specified, see Publishing And Rendering Options.
renderOptions	A string that specifies the options for the rendering modules. The default is **Nothing**. For information about the options that can be specified, see Publishing And Rendering Options.
userName	The name of the user who initiated the task. The user account must exist in the Meridian Enterprise Server account database as described in *Create And Edit User Accounts* in the *Meridian Enterprise Server Administrator's Guide*. If this parameter is **Nothing** or an empty string, the name of the current user is used.
preventDuplicate	A Boolean value indicating whether to check for duplicate items before adding a new one. If any duplicates are found, the item will not be created. If **Nothing**, then the option set for the publishing job is used.

Name	Description
feedbackProperty	The name of the property where to store the result of the registration.
customColumns	A 2D array of values (*<columnName>*, *<columnValue>*) that represent custom columns in the Publisher Queue database.

Return Value

An integer value that uniquely identifies the registered document in the queue.

Remarks

Before calling a **RegisterDocument** from VBScript, you should create it with the **AMCreateObject** function and specify the ProgId of the object as described in AMCreateObject Function. To explicitly delete and release the object, call the **Dispose** method.

Instantiating a **MeridianQueue** object to publish documents can take longer than publishing the same document with the PowerUser extension.

RevokeBatch Method

Revokes all documents that have been registered in the Publisher queue since the calling of the **BeginBatch** method.

Syntax

```
Function RevokeBatch (vaultId)
```

Parameters

Name	Description
vaultId	A string that uniquely identifies the vault in which the document resides. Specify **Nothing** or an empty string to work in the same vault where the VBScript runs. The vault identifier has the syntax **\\server\vault**, where **server** is the name of the server computer and **vault** is the internal name of the vault or datastore. The vault identifier may consist of only a vault name, in which case the vault's default server, if set, will be assumed.

Remarks

This method must be called in the **_Terminate** event procedure of a custom command, similar to the example shown in BeginBatch method. If this is not done, then the **RegisterDocument** method revokes documents in the queue individually.

MeridianTask Object

The Meridian Enterprise Task Server can be used to register documents for publication with the **MeridianTask** object properties that are described below.

MeridianTask Object Properties

The **MeridianTask** object provides the properties that are described below.

> **Note:**
> All unknown property assignments passed to the **BCPublisher.MeridianTask** object are considered as custom column names and values.

DocumentID Property

A value that uniquely identifies the document to publish. If **Nothing** (default) or an empty string, then the selected document is used.

Syntax

```
DocumentID As String
```

FeedbackProperty Property

The name of the property where to store the result of the document registration. The default is **Nothing**.

Syntax

```
FeedbackProperty As String
```

JobCode Property

A value that uniquely identifies the publishing job.

Syntax

```
JobCode As String
```

Remarks

If the job has multiple destination systems configured and the document should be published to only one of the systems, this string must specify the destination system display name. Use the syntax **jobId>systemDisplayName** (for example, **MyJob>MySystem**). If the document should be published to all of the destination systems, the system name should be omitted.

PreventDuplicate Property

A value indicating whether to check for duplicate items before to adding a new one. If any duplicates are found, the item will not be created. If **Nothing** (default), then the option set for the publishing job is used.

Syntax

```
PreventDuplicate As Boolean
```

PublishOptions Property

The publishing options for the current system links. Each system link defines the options that it supports. For information about the publishing options supported by a specific system link, see the system link description in *Publishing Modules* in the *Meridian Enterprise Server Administrator's Guide*. The options can be specified in the source system client application, if supported by the link, or they can be specified in VBScript event handlers. The default is **Nothing**.

Syntax

```
PublishOptions As String
```

RenderOptions Property

The options for the rendering modules. The default is **Nothing**.

Syntax

```
RenderOptions As String
```

RevisionID Property

A value that uniquely identifies the revision of the document. The default is **Nothing**.

Syntax

```
RevisionID As String
```

UserName Property

The name of the user who initiated the task. If **Nothing** (default) or an empty string, the name of the Task Server service account is used.

Syntax

```
UserName As String
```

NewMailMessage Object

The **NewMailMessage** object allows the sending of email messages using the MAPI interface of the user's registered email application. The **NewMailMessage** object supports full addressing and attachment capabilities. This functionality is the same as the **Send to Mail Recipient** command in the Meridian client.

NewMailMessage Object Properties

The **NewMailMessage** object provides the following properties.

Attachments Property

A collection of attachments for the current message. Read-only.

Syntax

```
Attachments As IASMailAttachments
```

NoteText Property

The body text of the current message.

Syntax

```
NoteText As String
```

Originator Property

An object that represents the originator of the message.

Syntax

```
Originator As IASMailRecipient
```

Recipients Property

A collection of recipients of the current message.

Syntax

```
Recipients As IASMailRecipients
```

Subject Property

The subject text of the current message.

Syntax

```
Subject As String
```

NewMailMessage Object Methods

The **NewMailMessage** object provides the methods described below.

Clean Method

Clears all **NewMailMessage** object properties.

Syntax

```
Clean()
```

Parameters

This method accepts no parameters.

Send Method

Sends the current **NewMailMessage** object.

Syntax

```
Send(Flags As Long)
```

Parameters

Name	Description
Flags	A combination of one or more **AS_MAPIMSG_SEND_FLAGS** constants. Read-only.

ProjectCopy Object

The **ProjectCopy** object represents a project copy document of the current **Document** object. The properties of the **ProjectCopy** object are described below.

ProjectCopy Object Properties

The **ProjectCopy** object provides the following properties.

IsActive Property

True if the project copy is active. Read-only.

Syntax

```
IsActive As Boolean
```

MasterLocked Property

True if the corresponding master document is locked. Read-only.

Syntax

```
MasterLocked As AS_PCLOCK
```

HasConcurrentPCs Property

True if the current **Document** object has multiple concurrent project copies. Read-only.

Syntax

```
HasConcurrentPCs As Boolean
```

ProjectCopy Object Methods

The **ProjectCopy** object provides the following methods.

UnlinkFromMaster Method

Removes the reference between a project copy and its master document and optionally unlocks the master document if it was locked by the project copy that was detached.

Syntax

```
UnlinkFromMaster([UnlockMaster As Boolean = True])
```

Parameters

Name	Description
UnlockMaster	If **True**, unlocks the master document after the reference has been deleted. Optional.

RequireMerge Method

Reverses the effects of the **Confirm Merged with Master** command and reactivates the project copy.

Syntax

```
Function RequireMerge()
```

Parameters

This method accepts no parameters.

UndoMakeObsolete Method

Reverses the effects of the **Confirm Superseded by Master** command and reactivates the project copy.

Syntax

```
Function UndoMakeObsolete()
```

Parameters

This method accepts no parameters.

Query Object

The **Query** object represents a lookup list query to an external database. The **Query** object is available to the **Vault** object.

Query Object Properties

The **Query** object provides the following properties, all of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

DBConnection Property

An ADO connection object. Read-only.

Syntax

```
DBConnection As Object
```

DisplayName Property

The display name of the query as seen by users. Read-only.

Syntax

```
DisplayName As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an

email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

Type Property

The type of the query as one of the **AS_TQTYPE** constants. Read-only.

Syntax

```
Type As AS_TQTYPE
```

Query Object Methods

The **Query** object methods affect the current **Query** object. They are described below.

GetValues Method

The **GetValues** method of the **Query** object is the same as the GetValues method of the **Table** object except that it operates on a **Query** object.

GetValuesEx Method

The **GetValuesEx** method of the **Query** object is the same as the GetValuesEx method of the **Table** object except that it operates on a **Query** object.

References Object

The **References** object represents the collection of Meridian references of a document. The **References** object is available to the current **Document** object.

> **Note:**
> The **References** object is accessed by the Document.GetReferences method.

References Object Properties

The **References** object provides the following properties, all of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

Count Property

The total number of references of the current **Document** object. Read-only.

Syntax

```
Count As Long
```

Exist Property

Returns **True** if the document specified by **DocID** is referenced by the current **Document**object.

Syntax

```
Exist(DocID) As Boolean
```

Properties Property

Returns an array of properties for the reference.

Syntax

```
Properties As Variant()
```

Return Value

The array contains the following information.

Note:
The values **Target/Source** and **Source/Target** in the **Object** column indicate that the object depends on the direction of the reference.

Return Values of Properties

Index Value	Object	Property	Notes
0		Internal Properties	This is an internal property that is not relevant to end-users.
1		Internal Properties	This is an internal property that is not relevant to end-users.
2	DocRef	InternalName	
3	DocRef	DisplayName	
4	DocRefType	ID	
5	Target/Source	Document ID	**Target/Source** in the **Object** column indicates that the object depends on the direction of the reference.
6	Target/Source	Document Display Name	**Target/Source** in the **Object** column indicates that the object depends on the direction of the reference.
7	Target/Source	Reference Description	**Target/Source** in the **Object** column indicates that the object depends on the direction of the reference.
8	Source/Target	Document ID	**Source/Target** in the **Object** column indicates that the object depends on the direction of the reference.
9	Source/Target	Document Display Name	**Source/Target** in the **Object** column indicates that the object depends on the direction of the reference.
10	Source/Target	Reference Description	**Source/Target** in the **Object** column indicates that the object depends on the direction of the reference.
11		DRT	This is an internal property that is not relevant to end-users.
12		DRT_OPTION_FLAGS	This is an internal property that is not relevant to end-users.
13	Reference	InternalName	The internal name of the reference.

Example

```
If Not Document Is Nothing Then
    'Add your code for document objects
    Dim props
    props = Document.GetReferences().Properties
    Dim aprop
    aprop = props(0)
    WinMsgBox aprop(13)
ElseIf Not Folder Is Nothing Then
    'Add your code for folder objects
End If
```

Target Property

Gets the referenced document specified by the collection index or **DocID** in **Item**. Read-only.

Syntax

```
Target(Item) As IASDocument5
```

References Object Methods

The **References** object methods affect the current **Document** object.

Add Method

Adds a reference by index or **DocID** to the current **Document** object.

Syntax

```
Add(DocID As String, [RefDisplayName As String])
```

Parameters

Name	Description
DocID	The **DocID** value of the document to reference.
RefDisplayName	The name to display for the reference as seen by users.

Remarks

For events to occur for this method, the **DocRefEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

Delete Method

Deletes a reference from the current **Document** object by **References** index number or **DocID**.

Syntax

`Delete(`*`Item`*`)`

Parameters

Name	Description
Item	The **References** index or **DocID** value of the reference to delete.

Remarks

For events to occur for this method, the **DocRefEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y** (default).

Refresh Method

Refreshes the current **References** collection.

Syntax

`Refresh`

Parameters

This method accepts no parameters.

Report Object

Use the **Report** object in configuration expressions to define a report template.

Report Object Properties

The **Report** object provides the following properties, all of which are read-only.

BriefCase Property

An object that represents a briefcase. If the parent object is a document, this is the briefcase to which the document is locked. If the parent object is a report (transmittal sheet), this is the briefcase that contains the report. Read-only.

Syntax

```
BriefCase As IASBriefCase
```

DisplayName Property

The name of the briefcase as seen by users. Read-only.

Syntax

```
DisplayName As String
```

DocumentCount Property

The number of documents contained in the report. Read-only.

Syntax

```
DocumentCount As Long
```

FileName Property

The file name (**Display Name** property) of the document or briefcase.

Syntax

```
FileName As String
```

Remarks

For events to occur when setting this property for a document, the **RenameEvents** setting in the **[ScriptEvents]** section on the **Application Integration** tab of **Application Settings** in Configurator must be set to **Y**. The default is **N**.

Format Property

The name of the template used by the report. Read-only.

Syntax

```
Format As String
```

QuoteChar Property

The text delimiter character used by the report. Read-only.

Syntax

```
QuoteChar As String
```

SepChar Property

The field separator character used by the report. Read-only.

Syntax

```
SepChar As String
```

VaultName Property

The name of the vault in which this report resides. Read-only.

Syntax

```
VaultName As String
```

Roles Object

The **Roles** object manages security roles for folders. The scope of the roles is determined by its parent folder.

Roles Object Properties

The **Roles** object provides the following properties, some of which are read-only. To affect some read-only properties requires user interaction or custom event procedures that use other **Roles** properties or methods.

Assignments Property

Gets the names of users and groups that have been assigned to roles for the current **Folder** object.

Syntax

```
Assignments As Variant
```

Remarks

Returns an array of subarrays. Each subarray contains the name of the user or group and its corresponding role name.

InheritedFromParent Property

Gets or sets whether the current **Folder** object inherits its role assignments from its parent object.

Syntax

```
InheritedFromParent As Boolean
```

Remarks

This property is set to **False** by the **AddAssignment**, **Clear**, and **DeleteAssignment** methods.

Roles Object Methods

The **Roles** object methods affect the current **Folder** object.

AddAssignment Method

Adds a role assignment to the current **Folder** object.

Syntax

```
AddAssignment (Assignee As String, Role As String)
```

Parameters

Name	Description
Assignee	User or group name to assign the role to.
Role	Role name to assign to **Assignee**.

Remarks

This method sets the **InheritedFromParent** property to **False**.

Clear Method

Removes all existing role assignments from the current **Folder** object.

Syntax

```
Clear
```

Parameters

This method accepts no parameters.

Remarks

This method sets the **InheritedFromParent** property to **False**.

DeleteAssignment Method

Removes a role assignment from the current **Folder** object.

Syntax

```
DeleteAssignment (Assignee As String, Role As String)
```

Parameters

Name	Description
Assignee	User or group name to remove the role from.
Role	Role name to remove from **Assignee**.

Remarks

This method sets the **InheritedFromParent** property to **False**.

Scope Object

The **Scope** object represents a named feature set that is configured for the current vault. A scope simplifies and limits users' access to only the configurable features that the users need.

For more information about scopes, see the *Scopes* section in the *Meridian Enterprise Configuration Guide*.

Scope Object Properties

The **Scope** object provides the following properties.

DisplayName Property

The **Display Name** property of the scope.

Syntax

```
Vault.Scope(<ScopeInternalName>).Displayname As String
```

Remarks

This property affects scopes defined in Meridian Enterprise Configurator, not scopes defined by the VaultEvent_ChangeScope event.

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with

Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

DocumentTypes Property

Returns a semicolon-delimited string containing a list of the document types that have been configured for the selected scope. Read-only.

Syntax

```
Vault.Scope(Client.CurrentScope).DocumentTypes As String
```

Remarks

Calling this method results in an error if the user has nothing selected.

RootFolder Property

The name of the root folder of the scope. Read-only.

Syntax

```
Vault.Scope(<ScopeInternalName>).RootFolder As String
```

Remarks

To obtain a folder object that is the root folder of the scope, use the expression:

```
Vault.RootFolder.GetSubFolder(Vault.Scope
(<ScopeInternalName>).RootFolder)
```

The **Scope** object provides the following methods.

Sequence Object

The **Sequence** object manages sequence numbers for a specific class of objects. Sequences are useful for generating unique numbers used in calculating document names. The scope of a sequence is determined by its parent object.

Sequence Object Properties

The **Sequence** object provides the following properties.

Next Property

Increment the current sequence value or start a new sequence at an optional specified value. Read-only. The value of **StartAt** must be higher than the current sequence value.

Syntax

```
Next(StartAt)As Long
```

Value Property

The current sequence value. Read-only.

Syntax

```
Value As Long
```

Sequence Object Methods

The **Sequence** object provides the methods described below.

SetTo Method

Set the current sequence to a new number.

Syntax

```
SetTo(Number)
```

Parameters

Name	Description
Number	Long integer with which to set the sequence.

Return Value

The new (Long) number.

StaticCollection Object

The **StaticCollection** object is returned by the StaticCollection property of the **Vault** object.

StaticCollection Object Methods

The following methods are available for the **StaticCollection** object.

Add Method

Adds a document to the static collection. You must specify the document object. If the document is already in the static collection, an error is returned.

Syntax

```
Add (Document)
```

Parameters

Name	Description
Document	A Document object.

Remove Method

Removes a document from the static collection. You must specify the document object.

Syntax

```
Remove(Document)
```

Parameters

Name	Description
Document	A Document object.

RemoveAll Method

Removes all documents from the static collection.

Syntax

```
RemoveAll ()
```

Parameters

This method accepts no parameters.

Table Object

The **Table** object represents a lookup list table in the current vault.

Table Object Properties

The **Table** object provides the following properties, all of which are read-only.

DBConnection Property

An ADO connection object. Read-only.

Syntax

```
DBConnection As Object
```

ColumnsInfo Property

A two-dimensional array of table data. Read-only.

Syntax

```
ColumnsInfo As Variant
```

DisplayName Property

The **Display Name** of the table as entered in Configurator. Read-only.

Syntax

```
DisplayName As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

Type Property

The type of the table as one of hte **AS_TQTYPE** constants. Read-only.

Syntax

```
Type As AS_TQTYPE
```

Table Object Methods

The **Table** object provides the methods described below.

AddValues Method

Adds values to the current **Table** object.

Syntax

```
AddValues([Columns], [Values])
```

Parameters

Name	Description
Columns	Optional names of columns to add to the table.
Values	Optional values to add to the table.

Clean Method

Removes all rows from the current **Table** object.

Syntax

```
Clean()
```

Parameters

This method accepts no parameters.

DeleteValues Method

Deletes values from the current **Table** object.

Syntax

```
DeleteValues([MatchColumns], [MatchValues])
```

Parameters

Name	Description
MatchColumns	Optional names of columns to delete from the table.
MatchValues	Optional values to delete from the table.

DeleteValuesEx Method

Deletes values from the current **Table** object.

Syntax

```
DeleteValuesEx(Where As String, [Parameters])
```

Parameters

Name	Description
Where	SQL expression that specifies the values to delete.
Parameters	Optional parameters.

GetValues Method

Gets values from the current **Table** object that match the specified criteria. If the query does not match any values, the method returns an empty variant.

Syntax

```
GetValues([MatchColumns], [MatchValues], [OutColumns], [Distinct],
[OrderBy], [FieldsInfo]) As Variant
```

Parameters

Name	Description
MatchColumns	Optional array of column names to match in the table.
MatchValues	Optional array of column values to match in the table. The index of each value should match the index of its corresponding column in **MatchColumns**.
OutColumns	Optional array of column names to return in the result.
Distinct	If **True**, returns only the unique values. Optional.
OrderBy	Optional array of column names with which to sort the returned values.
FieldsInfo	Optional variant array to contain information about the returned columns. The information includes: • **FieldsInfo(0)**: Name of the column in the recordset (AS_CI_NAME) • **FieldsInfo(1)**: Data type (AS_CI_SIZE) • **FieldsInfo(2)**: Size (AS_CI_TYPE) • **FieldsInfo(3)**: Column attributes (AS_CI_ATTR) For the values of possible data types, see DataTypeEnum in MSDN. For the values of possible column attributes, see FieldAttributeEnum in MSDN.

Return Value

A 2-dimension variant array with the property values in the second dimension of the array.

Remarks

The **GetValues** method executes an SQL query with the specified parameters. For example, the following VBScript statement:

```
Vault.Table("Employees").GetValues(Array("Role","Department"), _
    Array("Manager","Engineering"), _
```

```
    Array("FirstName","LastName"), _
    False, "LastName")
```

Translates into the following SQL query:

```
SELECT FirstName, LastName FROM Employees WHERE Role='Manager' AND
Department='Engineering' ORDER BY LastName
```

If the **FieldsInfo** parameter is specified, that array contains information about the returned value array. This can be useful if you want a generic function that can operate with the results of different queries regardless the output columns and the order in which order they were requested. Moreover, using **FieldsInfo,** you can process results from different tables (or queries to external data sources) without specific knowledge of the column names, which may be named differently in different tables but have the same purpose in the context of your processing.

The following example functions demonstrate the use of **FieldsInfo**.

```
' Helper function that finds the index of a ShareName column in the
results table.
Function FindShareNameIndex(fi)
  FindShareNameIndex = -1
  If IsArray(fi) Then
    For i = 0 To UBound(fi,1)
      If fi(0,i) = "ShareName" And fi(1,i) = 202 Then
        FindShareNameIndex = i
        Exit Function
      End If
    Next
  End If
End Function

Function TestFieldsInfo()
  Dim fi
  Dim res
  res = Vault.Table("GCFShares").GetValues(,,,,,fi)
  If IsArray(fi) Then
    Dim sni
    sni = FindShareNameIndex(fi)
    TestFieldsInfo = ""
    For j = 0 To UBound(res,2)
      TestFieldsInfo = TestFieldsInfo & res(sni,j) & ";"
    Next
  Else
    TestFieldsInfo = fi
  End If
End Function
```

GetValuesEx Method

Gets values from the current **Table** object that match the specified criteria. If the query does not match any values, the method returns an empty variant.

Syntax

```
GetValuesEx([SelectList As String], [Where As String],[GroupBy As
String], [Having As String], [OrderBy As String], _
[Parameters], [FieldsInfo]) As Variant
```

Parameters

Name	Description
SelectList	Optional comma-separated list of column names to retrieve values from the table.
Where	Optional condition for matching column values to retrieve from the table.
GroupBy	Optional column names to group values in the result.
Having	Optional aggregate function that is a condition for grouping returned values.
OrderBy	Optional comma-separated list of column names with which to sort the returned values.
Parameters	Optional value or array of values to be used in a parametric query. The values do not need to be strings. The array should contain a value for each instance of the ? character in the SQL query.
FieldsInfo	Optional variant array to contain information about the returned columns. The information includes: • **FieldsInfo(0)**: Name of the column in the recordset (AS_CI_NAME) • **FieldsInfo(1)**: Data type (AS_CI_SIZE) • **FieldsInfo(2)**: Size (AS_CI_TYPE) • **FieldsInfo(3)**: Column attributes (AS_CI_ATTR) For the values of possible data types, see http://msdn.microsoft.com/en-us/library/ms675318(VS.85).aspx. For the values of possible column attributes, see http://msdn.microsoft.com/en-us/library/ms676553(VS.85).aspx.

Return Value

A 2-dimension variant array with the property values in the second dimension of the array.

Remarks

The **GetValuesEx** method is functionally equivalent to **GetValues**. However, the **GetValuesEx** method supports parameters in the same order as a standard SQL query. This may be more convenient to use than **GetValues** if you are already familiar with the SQL language.

If the **FieldsInfo** parameter is specified, that array contains information about the returned value array. This can be useful if you want a generic function that can operate with the results of different queries regardless of the output columns and the order in which order they were requested. Moreover, using **FieldsInfo,** you can process results from different tables (or queries to external data sources) without specific knowledge of the column names, which may be named differently in different tables but have the same purpose in the context of your processing.

The following example functions demonstrate the use of **FieldsInfo**.

```
' Helper function that finds the index of a ShareName column in the
results table.
Function FindShareNameIndex(fi)
  FindShareNameIndex = -1
  If IsArray(fi) Then
    For i = 0 To UBound(fi,1)
      If fi(0,i) = "ShareName" And fi(1,i) = 202 Then
        FindShareNameIndex = i
        Exit Function
      End If
    Next
  End If
End Function

Function TestFieldsInfo()
  Dim fi
  Dim res
  res = Vault.Table("GCFShares").GetValues(,,,,,fi)
  If IsArray(fi) Then
    Dim sni
    sni = FindShareNameIndex(fi)
    TestFieldsInfo = ""
    For j = 0 To UBound(res,2)
      TestFieldsInfo = TestFieldsInfo & res(sni,j) & ";"
    Next
  Else
    TestFieldsInfo = fi
  End If
End Function
```

UpdateValues Method

Updates values in the current **Table** object.

Syntax

```
UpdateValues(MatchColumns, MatchValues, SetColumns, SetValues)
```

Parameters

Name	Description
MatchColumns	Array of matching column names to update in the current table.
MatchValues	Array of matching column values to update in the current table.
SetColumns	Array of column names to set in the current table.
SetValues	Array of column values to set in the current table.

Remarks

The **UpdateValues** method executes an SQL query with the specified parameters. For example, the following VBScript statement:

```
Vault.Table("Employees").UpdateValues(Array("FirstName","LastName"),
Array("John","Smith"), Array("Role", "Department"), Array("Manager",
"Engineering"))
```

translates into the following SQL query:

```
UPDATE Employees SET Role='Manager' AND Department='Engineering'
WHERE FirstName='John', LastName='Smith'
```

UpdateValuesEx Method

Updates values in the current **Table** object.

Syntax

```
UpdateValuesEx(Where As String, Parameters, SetColumns, SetValues)
```

Parameters

Name	Description
Where	Condition for matching column values to update in the current table.
Parameters	String or array of strings of values to be used in a parameterized query. The array should contain a value for each instance of the ? character in the SQL query.
SetColumns	Array of column names to set in the table.
SetValues	Array of column values to set in the table.

Remarks

The **UpdateValuesEx** method is functionally equivalent to **UpdateValues**. However, the **UpdateValuesEx** method supports parameters in the same order as a standard SQL query. This may be more convenient to use than **UpdateValues** if you are already familiar with the SQL language.

Task Object

The **Task** object sends a task request to a Accruent Task Server task extension for execution, such as to send an email. For more information about the Accruent Task Server, see *Meridian Task Server* in the *Meridian Enterprise Administrator's Guide* .

Task Object Methods

The **Vault.Task** object provides the methods described below.

Reset Method

Clears the arguments of the current **Task** object.

Syntax

```
Reset()
```

Parameters

This method accepts no parameters.

Remarks

Reset the **Task** object before each task.

Set Method

Sets the value of the specified argument.

Syntax

```
Set(Argument As String, Value)
```

Parameters

Name	Description
Argument	The name of the argument to set.
Value	The value to which to set the argument.

Remarks

The value of **Argument** is passed to the task extension on the Task Server as a parameter of the task to be performed. Each task extension defines its own arguments. Refer to the documentation of the task extension to find the appropriate argument names.

The argument names passed to Task Server extensions built with the Accruent .NET API are case-sensitive.

Submit Method

Submits the current **Task** object to the Task Server.

Syntax

```
Submit(Type As String, [FSObject As IUnknown*], [StartAt], [Priority
As Long = 0], [Server As String])
```

Parameters

Name	Description
Type	The type of task that will be executed. This must be the ProgID of the task extension that will perform the task. The ProgID is a combination of the task extension project name and the class name used by the Visual Basic project that produced the task extension.
FSObject	Optional **Document** or **Folder** object. This parameter is only required if the task extension uses this information.
StartAt	Optional date value in Greenwich Mean Time (GMT) until which to delay the execution of the task.
Priority	Reserved for future use.
Server	Optional name of the computer that is running the Task Server service. The task will be executed on this computer. You can omit this parameter to use the default Task Server computer.

Remarks

The argument names passed to Task Server extensions built with the Accruent .NET API are case-sensitive.

User object

The **User** object represents the current user and is available to all event procedures.

User Object Properties

The **User** object provides the following properties, all of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

> **Note:**
> Meridian user names can be shown in different formats as specified by the server registry setting **UserNameFormat** described in HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase in the *Meridian Enterprise Administrator's Guide*.

AllEmails Property

All email addresses of the user as stored in the user's Meridian account information. Read-only.

Syntax

```
AllEmails As String
```

Description Property

The description of the user as stored in the user's Meridian account information. Read-only.

Syntax

```
Description As String
```

EmailAddress Property

The default email address of the user as stored in the user's Meridian account information. Read-only.

Syntax

```
EmailAddress As String
```

FullName Property

The full name of the user as stored in the user's Meridian account information. Read-only.

> **Note:**
> Meridian user names can be shown in different formats as specified by the server registry setting **UserNameFormat** described in HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase in the *Meridian Enterprise Administrator's Guide*.

Syntax

```
FullName As String
```

HasPrivilege Property

True if the user has the specified privilege for the default object (**ParentFolder** or **WorkArea**) or an optional specified **Document** (expressed as **Document.ParentFolder**), **Vault**, **WorkArea**, or **Folder** object. The **Privilege** argument is specified as a long integer that represents one or more of the **AS_PRIVILEGES** constants. Read-only.

Syntax

```
HasPrivilege(Privilege, [Object]) As Boolean
```

HasRole Property

True if the user is assigned the specified role for the default object (**ParentFolder** or **WorkArea**) or an optional specified **Document** (expressed as **Document.ParentFolder**), **Folder**, **WorkArea**, or **Vault** object. Read-only.

Syntax

```
HasRole(Role, [Object]) As Boolean
```

Remarks

To use this property in a visibility expression to conditionally show custom properties or pages in a new document wizard, the **Object** parameter must be specified as the expected destination folder of the new document. By default, new documents do not yet have a **ParentFolder** value in the **DocGenericEvent_BeforeNewDocument** event when the wizard is active.

If the default or a specified object does not have any roles assigned and also does not inherit any role assignments from a parent, this property will always return True. This behavior is by design; role-based security is disabled if there are no roles assigned in the vault. To enable role-based security and return a different result, assign at least one role to the root of the vault.

Initials Property

The initials of the user as stored in the user's Meridian account information. Read-only.

Syntax

```
Initials As String
```

Manager Property

The manager of the user as stored in the user's Meridian account information. Read-only.

Syntax

```
Manager As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

OrganizationalUnit Property

The organizational unit of the user as stored in the user's Meridian account information. Read-only.

Syntax

```
OrganizationalUnit As String
```

Profile Property

The user's personal PowerWeb settings. Each setting represents the user interface options described in the *Personal preferences* article in the *Meridian Enterprise User's Guide*.

Syntax

```
Profile.ActiveXCompatibilityMode As Boolean
Profile.CurrentTimeZone As String
Profile.Language As String (case sensitive Windows Language Code Identifier)
Profile.Locale As String (case sensitive Windows Language Code Identifier)
Profile.SiteCacheMode As Boolean
Profile.ViewRenditions As Boolean
Profile.CreateThumbnailsOnImport As Boolean
Profile.UpdateThumbNailsAfterQuickChange As Boolean
Profile.RememberVaultLocation As Boolean
Profile.ShowRibbon As Boolean Profile.ColorScheme
```

Remarks

Set values in the VaultEvent_NewProfile event.

Title Property

The title of the user as stored in the user's Meridian account information. Read-only.

Syntax

```
Title As String
```

User Object Methods

The **User** object provides no methods. To modify user properties, edit the user's Meridian account as described in *Create and Edit User Accounts* in the *Meridian Enterprise Administrator's Guide* .

Vault Object

The **Vault** object represents the current vault and is available to all event procedures.

Vault Object Properties

The **Vault** object provides the following properties, most of which are read-only. To affect some read-only properties requires user interaction or custom event procedures.

Argument Property

A variable stored with the **Vault** object for as long as the current user has the vault open. The variable is local to the user and cannot be seen by other users.

Syntax

```
Argument(Name) As Variant
```

Remarks

The value of **Name** is case-sensitive.

Following are the predefined, read-only **Vault** object arguments that you can use to obtain additional information that may be useful when working with particular events.

Vault object arguments

Argument	Description
Vault.Argument("__ $$RelatedTransmittal")	A reference to a transmittal document that is related to a submittal. Useful in the **DocGenericEvent_BeforeNewDocument** event.
Vault.Argument("__ $$SubmittalSender")	A reference to the sender of a submittal. Useful in the **DocGenericEvent_ BeforeNewDocument** event.

For more information about arguments, see Object Arguments.

CurrentWorkArea Property

An object that represents the current work area context. Read-only. The work area feature has been deprecated.

Syntax

```
CurrentWorkArea As IASWorkArea
```

HasFeature Property

True if the specified feature is enabled for the current vault. The **Feature** argument is specified as a long integer that represents one or more of the **AS_FEATURES** constants. Read-only.

Syntax

```
HasFeature(Feature) As Boolean
```

Moment Property

When the vault is opened to a specific moment in the past, this property returns that date and time. Read-only.

For more information about vault history, see *Vault History* in the *Meridian Enterprise Configuration Guide*.

Syntax

```
Moment As Date/Time
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

Option Property

Gets a customization setting that is stored on the **Settings** page in the **Vault Settings** group in the **Environment** branch in Meridian Enterprise Configurator. Read only.

The option name is specified as **<SectionName>.<OptionName>** and must match the structure and names used on the **Settings** page. The supported structure is the same as used on the tabs of the **Application Link Settings** group in the **Environment** branch in Meridian Enterprise Configurator.

Syntax

```
Option As String
```

Path Property

The path of the vault expressed as **\\<Machine>\<DataStore>\<Section>**.

Syntax

```
Path As String
```

Query Property

An object that represents the specified query. Read-only.

Syntax

```
Query(Name) As IASQuery
```

RootFolder Property

An object that represents the root folder of the vault. Read-only.

For more information on folder objects, see Folder Object. To learn about the folder structure of a vault, see the *Field-Path Definition* section in the *Meridian Enterprise Configuration Guide*.

Syntax

```
RootFolder As IASFolder
```

RootURL Property

The PowerWeb URL of the root folder of the vault. Read-only.

Syntax

```
RootUrl As String
```

Scope Property

An object that represents the scope with a given name.

Syntax

```
Scope(ScopeName As String) As IASScope
```

Sequence Property

An object that represents the specified vault sequence.

> **Important!**
> Using the vault name as a sequence name can cause errors and possibly crash the Meridian EDM Server service.

For information about the **Sequence** object, see Sequence Object. For information about using sequences in file names, see *File Name Calculation* in the *Meridian Enterprise Configuration Guide*.

Syntax

```
Sequence(Name) As IASSequence
```

ServerName Property

The name of the server hosting the current vault. Read-only.

Syntax

```
ServerName As String
```

ServerProductCode Property

A number expressed as a string that represents the server product. Read-only.

Syntax

```
ServerProductCode As String
```

ServerTimeGMT Property

The system time of the server computer expressed in Greenwich Mean Time (GMT). Read-only.

Syntax

```
ServerTimeGMT As Date
```

StaticCollection Property

This property can be used to get access to a static collection, or create a static collection.

Syntax

```
StaticCollection(CollectionName, [Create]) As IASStaticCollection
```

Parameters

Name	Description
CollectionName	The name of the static collection.
Create	This parameter allows you to create a new static collection.

Remarks

If you assign the result of this property to a variable, then you must set the variable to `Nothing` in the same subroutine or function.

Table Property

An object that represents the specified table. Read-only.

For information about tables, see *Create And Edit Tables* in the *Meridian Enterprise Configuration Guide*.

Syntax

```
Table(Name) As IASTable
```

Task Property

An object that represents the current **Task** object. Read-only.

For more information about task objects, see Task Object.

Syntax

```
Task As IASTask
```

User(Name) Property

An object that represents the specified user. Read-only.

For more information about user objects, see User object .

Syntax

```
User(Name) As IASUser
```

WebAccessLocationID Property

The PowerWeb URL of the vault. Read-only.

Syntax

```
WebAccessLocationID As String
```

WorkIsolationMode Property

True if the Work Isolation Mode of the current vault is enabled. Read-only.

Work Isolation Mode is a legacy feature. We still support customers who have it enabled, but we do not allow it to be enabled in new Vaults.

Syntax

```
WorkIsolationMode As Boolean
```

Vault Object Methods

The **Vault** object provides the methods described below.

AuditEvent Method

Adds custom command events to the audit log database.

Syntax

```
Sub AuditEvent(EventName As String, [ObjectName As String],
[ActionArg1 As String], [ActionArg2 As String], [ActionArg3 As
String])
```

Parameters

Name	Description
EventName	The name of the event that you want to add to the audit log database.
ObjectName	Optional name of the vault object upon which the custom event occurs.
ActionArg1	Optional argument that describes the action taken.
ActionArg2	Optional argument that describes the action taken.
ActionArg3	Optional argument that describes the action taken.

Remarks

Requires a Meridian FDA Module server license.

The object specified by **ObjectName** must not be Nothing. The optional action arguments can be anything that you decide. Each argument should be one parameter of the action taken and should resemble the general pattern used by the built-in audit entries listed in the *Audited Actions* article in the *Meridian Enterprise Administrator's Guide*.

CallRemote Method

Executes a remote procedure call.

Syntax

```
CallRemote(URL As String, Username As String, _
       Password As String, RemoteVault As String, _
       Script As String, [Args], [Flags As Long = 2])
```

Parameters

Name	Description
URL	Location of the server with which to connect.
Username	User account to use to connect to the server.
Password	Password of the user account.
RemoteVault	Name of the remote vault with which to connect.
Script	Name of the procedure to execute on the remote server.
Args	Optional arguments for the procedure to execute.
Flags	Optional flags from the **AS_CALLREMOTE_FLAGSS** enumeration.

Example

The following examples show how to use this method.

Example function defined in the called vault:

```
Function RemoteTest (First, Second, Third)
  RemoteTest = "RemoteTest returns: " & First & ", " & Second & ", "
& Third
End Function
```

Example procedure defined in the calling vault:

```
Sub Test_Execute(Batch)
  vArg = Array ("One ", "Two", "Three")
  vRes = Vault.CallRemote ("http://MyServer/Meridian", "MyUserName",

_
         "MyPassword", "MyVaultName", "RemoteTest", VArg, AS_CRF_
MULTIARGS)
  WinMsgBox vRes
End Sub
```

ComposeURL Method

Gets a PowerWeb URL for the current vault.

Syntax

```
ComposeURL(Object As Object, [Flags As AS_URL_FLAGS = AS_URL_
FULLPATH])
```

Parameters

Name	Description
Object	An object for which to return the address.
Flags	Optional long integer that represents one or more **AS_URL_FLAGS** constants.

Return Value

A string URL.

Remarks

The icon shown by shortcuts created with the return value of this method and the command line to execute those shortcuts are configured with the following registry keys that are described in the *Meridian Enterprise Administrator's Guide*.

```
HKEY_CLASSES_ROOT\BlueCielo
HKEY_CLASSES_ROOT\BlueCielo\DefaultIcon
HKEY_CLASSES_ROOT\BlueCielo\shell\open\command
```

The URL for the rendition of a document can be calculated by appending **;rend** to the URL of the main document as in the following example:

```
http://<ServerName>/Meridian/<VaultName>/MyDocument.doc;rend
```

CreateNewDocument

Creates a new document in the vault. Option to add references between new documents and update properties for the new document object.

Calling the `Vault.CreateNewDocument` method triggers the `Vault_Open` event in script.

Syntax

```
Vault.CreateNewDocument(documentPath As String, fileName As String,
docTypeName As String, templateName As String)
```

Parameters

Name	Description
documentPath	The location where the document should be created. This path should not include the file name.
fileName	Name of the new document.
docTypeName	Optional parameter which specifies the document type.
templateName	Optional parameter which specifies the name of the template configured for the document type.

Return Value

A new document object.

Example

```
Sub DocGenericEvent_AfterNewDocument(Batch, Action, SourceFile,
DocType, DocTemplate)
    If Document.DocumentType = "DocWithObject" And Action = AS_IT_
CREATED Then
        Dim newDoc
      Set newDoc = Vault.CreateNewDocument("\AT_4_
TagOperations\Objects","New.tag","Object","ObjectTemplate")
        Document.GetReferences("DocObjectReference").Add
(newDoc.GlobalID)
        newDoc.Property("psObject.ObjectDescription") = "New Created
object"
        newDoc.Property("psObject.ObjectType") = "ObjectType-01"
        newDoc.Property("psObject.ObjectNumber") = "ObjectNumber-01"
    End If
End Sub
```

DisplayName Method

Returns the display name of a specified scope.

Syntax

```
Vault(Scope As String).DisplayName
```

Parameters

Name	Description
Scope	The name of the scope for which to return the display name.

Return Value

The display name of the specified scope.

ExecSQL Method

Executes an SQL query using the specified connection string.

Syntax

```
ExecSQL(Connection As String, Query As String, [Parameters])
```

Parameters

Name	Description
Connection	A valid connection string for the database with which to execute the query. This parameter also accepts a variable that represents one of the following vault databases. The variables are case-sensitive. **$$TransManDB**: The Transmittal Management Module database (if installed) **$$UserDB**: The user account database When the **Use Enterprise Server for user management** option is enabled in the **EDM Server** properties in the Meridian Enterprise Administrator, the reserved word **$$UserDB** cannot be used in VBScript with the **Vault.ExecSQL** method to access the Meridian Enterprise Server user database. **$$<QueryName>**: External table query connection string created as described in *Create And Edit External Data Queries* in the *Meridian Enterprise Configuration Guide*
Query	A valid SQL query expression.
Parameters	Optional value or array of values to be used in a parametric query. The values do not need to be strings. The array should contain a value for each instance of the ? character in the SQL query.

FindDocuments Method

Searches for documents in the Main area of the current vault.

Syntax

```
FindDocuments([WildCard As String], [DocumentTypeNames As Variant],
[Criteria As Variant], OrSearch As Boolean) _
As IASDocuments
```

Parameters

Name	Description
WildCard	Optional string that represents a file system wildcard pattern. If omitted, the scope of file names will include all documents (*.*).
DocumentTypeNames	Optional variant array of document type names to which to restrict the search result. If omitted, the scope of document types will include all document types.
Criteria	Optional variant array of property filter criteria.
OrSearch	Optional Boolean that if set to **True** specifies a Boolean OR search be performed. The default is **False**.

Return Value

Returns a collection of Document Object objects matching the specified parameters.

Remarks

The **Criteria** parameter can be specified as a single criterion array or as an array of criterion arrays. Each criterion array contains a property name, operator, and an optional value similar to the **Find** command in PowerUser. For example, **Custom.ProjectNr, equals, 2134**, where **equals** is the operator.

The search operators that are supported are those in the **IC_OPERATOR** constants enumeration. The following table lists the supported operators and the abbreviations, symbols, and constants that may be used:

Search operator options

Operator	Abbreviation	Symbol	Constant
contains, in	C	*	IC_OP_CONTAINS
equals			IC_OP_DATE_EQUALS
not equal			IC_OP_DATE_NOT_EQUAL
doswildcard			IC_OP_DOSWILDCARD
empty	EM	()	IC_OP_EMPTY
equals, equal	E	EQ =	IC_OP_EQUALS
less	L	<	IC_OP_LESS
less than or equal	LE	<=	IC_OP_LESS_EQUAL
like	LI	%	IC_OP_LIKE
more than	M	>	IC_OP_MORE
more than or equal	ME	>=	IC_OP_MORE_EQUAL
not contains	NC, NOT IN		IC_OP_NOT_CONTAINS
not empty	NEM		IC_OP_NOT_EMPTY
not equal	NEQ, NE	<>	IC_OP_NOT_EQUAL
not like	NLI, NL		IC_OP_NOT_LIKE
not starts with	NSW		IC_OP_NOT_STARTSWITH
starts with	SW		IC_OP_STARTSWITH

- The expression `Vault.FindDocuments(<Criteria>).Count` will return the number of documents that are found.
- The expression `Vault.FindDocuments(<Criteria>).Properties` will return 25 values for each document found without the property names, in no particular order.
- The expression `Vault.FindDocuments(<Criteria>).Document(Document.ID).FileName` will return the file name of a document if it matches the specified parameters.
- If you want to find a document with a specific ID, use the **Vault.GetDocument** method described in GetDocument method instead. Executing an expression such as **Vault.FindDocuments.Document(ID)** will cause the server to first create a collection of all documents in the vault and then search within that collection for the document with the

specified ID. Such an operation can result in a large load on the server and take an excessive amount of time compared to using the **Vault.GetDocument** method.

- For best results when searching on date (day without time) values, use the operators with names that begin with **IC_OP_DATE**. For date with time (moment) searches, use the other operators.
- This method can return different results if it is called within the **Document_AfterNewDocument** event depending on whether Hypertrieve 3 or Hypertrieve 5 is used as the database engine of the vault. If the new document name matches the **WildCard** parameter, the collection of document IDs returned by a Hypertrieve 5 vault will contain the new document. A Hypertrieve 3 vault will not contain the new document. To make your script compatible with both vault types, it should filter the ID of the new document out of the search results. Similarly, if no results are expected (such as testing for a unique document name) but one document is returned, test whether it is the new document and proceed accordingly.

Example

The following example demonstrates use of the **FindDocuments** method to display the results of a search executed with the specified parameters.

```
Sub SearchCount_Execute(Batch)
    Dim StrMask
    StrMask = "*1*.*"

    Dim MultipleDocTypes
    MultipleDocTypes = Array("My_Doc_Type", "Hybrid_Doc_Type")

    Dim MultipleCriteria
    MultipleCriteria = Array(_
        Array("Custom.CI",  IC_OP_MORE, 200),_
        Array("Custom.CI",  IC_OP_LESS, 300),_
        Array("Custom.CS",  IC_OP_CONTAINS, "o"),_
        Array("Custom.CStr",  IC_OP_CONTAINS, "iv"),_
        Array("Custom.LOL",  IC_OP_EQUALS, "a-z"),_
        Array("AMVersionablePropertySet._VERSIONNUMBER",  IC_OP_LESS,
2),_
        Array("Custom.CDT",  IC_OP_LESS, (DateSerial(1983,07,12))),_
        Array("NewPS.CB",  IC_OP_EQUALS, False))

    Dim bool
    bool = True

    Dim Str
    Str = ""

    Dim Document
```

```
    For Each Document In Vault.FindDocuments(StrMask,
MultipleDocTypes, _
        MultipleCriteria, bool)
        str = str &_
            " FileName - > " & Document.FileName & vbCrLf &_
            " Document.DocumentType.DisplayName - > " & _
                Document.DocumentType.DisplayName & vbCrLf &_
            " Custom.CI - > " & Document.CI & vbCrLf &_
            " Custom.CS - > " & Document.CS & vbCrLf &_
            " Custom.CStr - > " & Document.CStr & vbCrLf &_
            " Custom.CDT - > " & Document.CDT & vbCrLf &_
            " NewPS.CB - > " & Document.NewPS_CB & vbCrLf &_
            " NewPS.CM - > " & Document.NewPS_CM & vbCrLf &_
            " Custom.LOL  - > " & Document.Custom_LOL & vbCrLf &_
            " Revision - > " & Document.Revision & vbCrLf & vbCrLf
    Next

    WinMsgBox "Total number of documents found is " &_
        Vault.FindDocuments(StrMask, MultipleDocTypes,
MultipleCriteria, bool).Count & _
            vbCrLf & vbCrLf & Str
End Sub
```

GetDistinctValues Method

Returns the unique values of the specified property.

Syntax

```
GetDistinctValues(PropertyName As String, MaxValues As Long = -1)
```

Parameters

Name	Description
PropertyName	The name of the property for which to retrieve values.
MaxValues	The maximum number of values to return. The default of **-1** returns all values.

Return Value

An array of the values for the specified property.

GetDocument Method

Returns the document object for a specified ID.

Syntax

```
GetDocument(ID As String) As IASDocument
```

Parameters

Name	Description
ID	Global ID of the document to retrieve.

Return Value

A **Document** object.

GetGroups Method

Returns the names of the current Meridian user groups. For more information about user groups, see *Create and Edit User Groups* in the *Meridian Enterprise Administrator's Guide*.

Syntax

```
GetGroups(GroupColumns As Long = 0, [User As String])
```

Parameters

Name	Description
GroupColumns	The number of columns of group information to retrieve. The default of **0** returns all columns.
User	Optional name of the user for which to return groups that the user is a member of.

Return Value

Returns an array of user group information. The dimensions of the array match the number of columns specified for **GroupColumns**.

GetPropertyNames Method

Get the property names for the specified property set.

Syntax

```
GetPropertyNames(PropertySetName As String) As Array
```

Parameters

Name	Description
PropertySetName	Name of the property set from which to return property names.

Return Value

A string array of property names.

GetUsers Method

Returns the names of the current Meridian user accounts. For more information about user accounts, see *Create and Edit User Accounts* in the *Meridian Enterprise Administrator's Guide*.

Syntax

```
GetUsers(UserColumns As Long = 0, [Group As String])
```

Parameters

Name	Description
UserColumns	The number of columns of user information to retrieve. The default of **0** returns all columns.
Group	Optional name of the group for which to return users that are members.

Return Value

Returns an array of user group information. The dimensions of the array match the number of columns specified for **UserColumns**.

RootFolder Method

Returns the root folder of a specified scope.

Syntax

```
Vault(Scope As String).RootFolder
```

Parameters

Name	Description
Scope	The name of the scope for which to return the root folder name.

Return Value

The root folder of the specified scope.

RunShellCommand Method

Opens a window and passes a specified command line to the operating system to execute. The command may be executed on the application server.

Syntax

```
RunShellCommand(Command As String, [ShowWindow As IC_SHOWWINDOW As
IC_SW_SHOWNORMAL], [AtServer As Boolean = False])
```

Parameters

Name	Description
Command	The command line to execute.
ShowWindow	Optional window style specified as one of the **IC_SHOWWINDOW** constants.
AtServer	Optional flag to run the command on the server.

SendNotification Method

Sends a notification email message for a specified object.

Syntax

```
SendNotification (NotificationName As String, Object As Object)
```

Parameters

Name	Description
NotificationName	The internal name of a notification definition created in Meridian Enterprise Configurator. The internal name can be seen in the tooltip that appears when the mouse cursor is hovered over the notification name.
Object	The object about which to send the notification, typically a document object.

Remarks

For information about creating notification definitions, see *Configure Event Notifications* in the *Meridian Enterprise Configuration Guide*.

Viewer Object

The **Viewer** object represents the Meridian viewer in the client applications.

Viewer Object Properties

The **Viewer** object provides the following properties.

IsPreview Property

True if the current document is being shown in the **Print Preview** window. Read-only.

Syntax

```
IsPreview As Boolean
```

Watermark Property

An object that represents the current watermark settings. For information on configuring watermark printing with the **Watermark** properties, see *Configure Watermark Printing* in the *Meridian Enterprise Configuration Guide*. This property supports two lines of text.

Syntax

```
Watermark As IASWatermark
```

Example

To achieve two lines, you can use a line break:

```
Client.Viewer.WaterMark.TitleText = "Printed by: " & User.FullName  +
vbCrLf + " on: " & DateValue(Today)
```

IsRendition Property

Indicates whether the viewed file is a rendition or the native content stream of the current document. Read-only.

Syntax

```
IsRendition As Boolean
```

WaitingList Object

The **WaitingList** object represents the waiting list that the current **Document** object is associated with. The properties of the **WaitingList** object are described below.

WaitingList Object Properties

The **WaitingList** object provides the following properties.

Documents Property

All project copies in the waiting list that the current document belongs to. Read-only.

Syntax

```
Documents As IASDocuments
```

Count Property

Number of documents in the waiting list. Read-only.

Syntax

```
Count As Long
```

Document Property

Allows you to get a document by index or id. Read-only.

Syntax

```
Document As IASDocument15
```

Property Property

Gets or sets the value of the specified (String) document property.

Syntax

```
Property(Name) As Variant
```

Priority Property

Priority of the project copy in the waiting list. Read-only.

Syntax

```
Priority As Long
```

WaitingList Object Methods

The **WaitingList** object provides the following methods.

Move Method

Changes the priority of a project copy in the waiting list for a document.

Syntax

```
Move([ProjectCopy As IASDocument15],[shiftValue As Long])
```

Parameters

Name	Description
ProjectCopy	The project copy that the priority is to be changed for.
shiftValue	The waiting list priority that you want to set for the document.

Remove Method

Removes the project copy from the waiting list.

Syntax

```
Move([ProjectCopy As IASDocument15])
```

Parameters

Name	Description
ProjectCopy	The project copy to be removed from the waiting list.

Workflow Object

The **Workflow** object represents a workflow definition. The **Workflow** object is available to the **Document** and **Folder** (if the Advanced Project Workflow Module is installed) objects.

Workflow Object Properties

The **Workflow** object provides following properties, all of which are read-only.

DisplayName Property

The name of the workflow as seen by users. Read-only.

Syntax

```
DisplayName As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

State Property

An object that represents the specified workflow state. Read-only.

Syntax

```
State(StateName) As IASWorkflowState
```

Transition Property

An object that represents the specified workflow transition. Read-only.

Syntax

```
Transition(TransitionName) As IASWorkflowState
```

Remarks

The **Transition** object provides a **SignatureRequired** property that can be used to determine if the transition requires electronic signatures.

- The property is **False** if the electronic signature feature is disabled or if it is enabled but the transition does not require electronic signatures.
- The property is **True** if the feature is enable and the transition requires one or more electronic signatures.

WorkflowState Object

The **WorkflowState** object represents a state in a workflow definition. The **WorkflowState** object is available to the **Document** and **Folder** (if the Advanced Project Workflow Module is installed) objects.

WorkflowState Object Properties

The **WorkflowState** object provides the following properties, all of which are read-only.

DisplayName Property

The name of the workflow state as seen by users. Read-only.

Syntax

```
DisplayName As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

Type Property

The type of the workflow state expressed as one of the **AS_WORKFLOW_STATE_TYPE** constants. Read-only.

Syntax

```
Type As AS_WORKFLOW_STATE_TYPE
```

Workflow Property

The parent workflow object. Read-only.

Syntax

```
Workflow As IASWorkflow
```

WorkflowTransition Object

The **WorkflowTransition** object represents a transition in a workflow definition. The **WorkflowTransition** is available to the **Document** and **Folder** objects if the Advanced Project Workflow Module is installed.

WorkflowTransition Object Properties

The **WorkflowTransition** object provides the following properties, all of which are read-only.

CreateNewRevision Property

If **True**, this transition creates a new revision of a document.

Syntax

```
CreateNewRevision As Boolean
```

DisplayName Property

The name of the workflow transition as seen by users. Read-only.

Syntax

```
DisplayName As String
```

Name Property

Returns the name of the current object. Depending on the object type, this property returns the display name of the object or it returns the internal name and may be read-only.

Syntax

```
Name As String
```

Remarks

If the current object is a **Package**, it must be in the **Open** status to set the value.

If the current object is a **User**, this property is the short name of the user as stored in the user's Meridian account information and is read-only. This property can be used to specify the recipient's email address and either an empty string or no value set in the **Address** property. If an

email address is specified in **Name**, it must be surrounded with carets (<>). When used with Microsoft Outlook, the **Name** property may contain the user's full name or last name and Outlook will attempt to resolve the name to the email address in the default address book.

Meridian user names can be shown in different formats determined by the **UserNameFormat** server registry setting, as described in the *HKEY_LOCAL_MACHINE\Software\Cyco\AutoManager Meridian\CurrentVersion\Server\UserDatabase* article in the *Meridian Enterprise Administrator's Guide*.

SourceState Property

The source state of the transition. Read-only.

Syntax

```
SourceState As IASWorkflowState
```

TargetState Property

The target state of the transition. Read-only.

Syntax

```
TargetState As IASWorkflowState
```

Workflow property

The parent workflow definition. Read-only.

Syntax

```
Workflow As IASWorkFlow
```

Meridian Functions

Meridian provides a number of functions in VBScript that you can use with Meridian objects and their properties. The functions are listed in the **Object Browser** in the Meridian Enterprise Script Editor.

Note:

- Some of the Meridian functions accept as parameters one or more predefined Meridian constants. The available Meridian constants can be found in the **Object Browser** in the Meridian Enterprise Script Editor.
- The functions that are provided by the Meridian Enterprise Asset Management Module are not listed in the Meridian Enterprise Script Editor.

AIMS_Commands Function

Adds up to three custom Meridian Enterprise command buttons to the Meridian Explorer feedback page.

Syntax

```
Function AIMS_Commands() As Array
```

Parameters

This function accepts no parameters.

Remarks

Each command button can run a VBScript function in the context of the current document in the Meridian Enterprise vault as demonstrated in the following example. The function must return an array of arrays that each define the command button caption and the name of a function to invoke when the button is clicked.

Example

```
Function AIMS_Commands()
    'For each button, provide a caption and function name in an array
    btnA = Array("Command A", "Do_ButtonA")
    btnB = Array("Command B", "Do_ButtonB")
    btnC = Array("Command C", "Do_ButtonC")

    'Return the array of commands
    AIMS_Commands= Array(btnA , btnB , btnC)
End Function

Sub Do_ButtonA()
    Document.Area1= "A"
    Document.ChangeManagementRemarks = "Option A chosen"
    Document.Log "Button A clicked at : " & CStr(Vault.ServerTimeGMT)
End Sub
```

AIMS_Properties Function

Invoked when a Meridian Explorer detail page is shown that has the **Enable feedback functionality** option enabled.

Syntax

```
Function AIMS_Properties()
```

Parameters

This function accepts no parameters.

Remarks

Use this method to work with the feedback page property information as demonstrated in the following example.

Example

```
Function AIMS_Properties()
    ' For each property, return its caption and value in an array
    p1 = Array("Asset description", Document.Area1)
    p2 = Array("My name", "Joe Engineer")

    'Return the array of properties
    AIMS_Properties = Array(p1, p2)
End Function
```

AIMS_UpdateChangeManagement Function

Invoked when a user saves changes to a Meridian Explorer detail page that has the **Enable feedback functionality** option enabled.

Syntax

```
Function AIMS_UpdateChangeManagement (Remarks As String, RequestType
As String)
```

Parameters

Name	Description
Remarks	The value that the user entered for the **Remarks** (**Custom.ChangeManagementRemarks**) property.
RequestType	The value that the user entered for the **Request type** (**Custom.ChangeManagementRequestType**) property.

Remarks

Use this function to modify the values that are saved by the user's input or to set other properties as demonstrated in the following example.

Example

```
Function AIMS_UpdateChangeManagement(remarks, requestType)
    Document.ChangeManagementRemarks = remarks
    Document.ChangeManagementRequestType = requestType
    Document.ApplyPropertyValues

    Call Document.Log(CStr(User.Name + " : " + requestType))

    AIMS_UpdateChangeManagement = ""
End Function
```

AMCreateObject Function

Creates and returns a reference to an object that is provided by an automation server. For more information about using the **AMCreateObject** function, see Automation Objects.

Syntax

```
AMCreateObject(ProgID As String, [OnServer As Boolean = False]) As
Object
```

Parameters

Name	Description
ProgID	A combination of the automation server name and the class name of the object to create.
OnServer	If this parameter is **True**, the object is created on the Meridian server instead of on the client computer. This parameter is meaningless in PowerWeb because the server is also the client.

Remarks

The normal VBScript **CreateObject** function is suitable for creating automation objects that do not act upon Meridian objects. For access to Meridian objects, use **AMCreateObject** instead, which works the same as **CreateObject** but also passes the current **Repository** object to the **IAMCommandSetInitialize** interface of the automation server.

AMMGetCustomColumnHeaders Function

Returns an array of column names that are displayed in the **Drawing Tags** dialog box.

Syntax

```
Function AMMGetCustomColumnHeaders() As String
```

Parameters

This function accepts no parameters.

Example

```
Function AMMGetCustomColumnHeaders()
    Dim mheaders(1)
    mheaders(0) = "Type Name"
    AMMGetCustomColumnHeaders = mheaders
End Function
```

Remarks

Called once during dialog box initialization.

AMMGetCustomColumnValues Function

Returns an array of column values for the **Drawing Tags** dialog box.

Syntax

```
Function AMMGetCustomColumnValues(TagId As String, Status As Byte, VaultObjectProperties As String,
DrawingObjectProperties As String) As String
```

Parameters

Name	Description
TagId	Tag number.
Status	Tag reference status described in the following table.
VaultObjectProperties	Array of the property values for the Meridian Enterprise object in the form `Array(AMObjectId, Referenced)`. The dimensions of the array are as follows: • `AMObjectId`: String value that uniquely identifies the object in Meridian Enterprise. If the object is not found in Meridian Enterprise, the value is Nothing • `Referenced`: Boolean value indicating whether the object has a tag reference

Name	Description
DrawingObjectProperties	Array of the property values for the asset in the form `Array(TagType, X, Y, LayerName, PnIDProperties)`. The dimensions of the array are as follows: • `TagType`: The string name of the tag type • `X`: String that contains the X coordinate provided by **PnIDLink_GetAssetCoordinates** • `Y`: String that contains the Y coordinate provided by **PnIDLink_GetAssetCoordinates** • `LayerName`: The string name of the layer on which the asset exists • `PnIDProperties`: Array of property values for the asset in the form `Array(PnIDProperty1, … PnIDPropertyN)`. The names of the properties are returned by the AMMPropertiesToBeRequested Function.

Status parameter byte values

Value	Status	Vault Object Exists	Drawing Object Exists	Reference Exists	Tag Type of Vault Object is Matched with One of Drawing Object	Vault Objects with Duplicate Tag Values Exist	Drawing Objects with Duplicate Tag Values Exist	Comment
0	NotDefined	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available	Status is not calculated yet. This status cannot be sent during a correct request.

Value	Status	Vault Object Exists	Drawing Object Exists	Reference Exists	Tag Type of Vault Object is Matched with One of Drawing Object	Vault Objects with Duplicate Tag Values Exist	Drawing Objects with Duplicate Tag Values Exist	Comment
1	Ok	Yes	Yes	Yes	Yes	No	No	
2	NotLinked	Yes	Yes	No	Yes	No	No	
3	OnlyInDocument	No	Yes	Not Available	Not Available	Not Available	No	
4	MismatchFound	Yes	Yes	No	No	No	No	
5	MismatchReferenced	Yes	Yes	Yes	No	No	No	
6	CoercedReference	Yes	No	Yes	Not Available	No	Not Available	
7	OnlyInVault	Yes	No	No	Not Available	No	Not Available	Object with this status exists just after unlink vault object with CoercedReference status. If the object will not be relinked (by **Link** or **Assign**) then the row will be deleted on the next refresh.

Value	Status	Vault Object Exists	Drawing Object Exists	Reference Exists	Tag Type of Vault Object is Matched with One of Drawing Object	Vault Objects with Duplicate Tag Values Exist	Drawing Objects with Duplicate Tag Values Exist	Comment
8	DuplicatesInVault	Yes	condition is not checked	condition is not checked	condition is not checked	Yes	No	
9	DuplicatesInDrawing	condition is not checked	Yes	condition is not checked	condition is not checked	No	Yes	
10	Duplicates	Yes	Yes	condition is not checked	condition is not checked	Yes	Yes	
254	Error	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available	Status calculation failed for any reason.

Example

```
Function AMMGetCustomColumnValues(TagId, Status, VaultObjectProperties, DrawingObjectProperties)
 Dim mCustomColumnValues(1)
    Dim mArray
    If Not IsNull(DrawingObjectProperties) And IsArray(DrawingObjectProperties) Then
        mArray = DrawingObjectProperties(4)
        If Not IsNull(mArray) And IsArray(mArray) Then
            mCustomColumnValues(0) = mArray(0)
```

```
            AMMGetCustomColumnValues = mCustomColumnValues
        End If
    End If
End Function
```

Remarks

Called for every asset in the drawing.

AMMGetReportCustomHeaderValues Function

Returns values to be placed in the header and footer area of the Excel report.

Syntax

```
Function AMMGetReportCustomHeaderValues (AccumulatedData As Variant)
As Variant
```

Parameters

Name	Description
AccumulatedData	A variant that contains user-defined data that can be accumulated from call to call and passed from the AMMGetReportTableRowValues Function.

Remarks

Called once after the last call to **AMMGetReportTableRowValues** that provides the **AccumulatedData** parameter.

AMMGetReportTableRowValues Function

Returns the array of column values for the Excel report.

Syntax

```
Function AMMGetReportTableRowValues(TagId As String, Status As Byte, VaultObjectProperties As String,
DrawingObjectProperties As String, AccumulatedData As Variant) As String
```

Parameters

Name	Description
TagId	Tag number
Status	Tag reference status described in the following table
VaultObjectProperties	Array of the property values for the Meridian Enterprise object in the form `Array(AMObjectId, Referenced)`. The dimensions of the array are as follows: • `AMObjectId`: A string value that uniquely identifies the object in Meridian Enterprise. If the object is not found in Meridian Enterprise, the value is Nothing. • `Referenced`: A Boolean value indicating whether the object has a tag reference.

Name	Description
DrawingObjectProperties	Array of the property values for the asset in the form `Array(TagType, X, Y, LayerName, PnIDProperties)`. The dimensions of the array are as follows: • `TagType`: The string name of the tag type • `X`: A string that contains the X coordinate provided by **PnIDLink_GetAssetCoordinates** • `Y`: A string that contains the Y coordinate provided by **PnIDLink_GetAssetCoordinates** • `LayerName`: The string name of the layer on which the asset exists • `PnIDProperties`: An array of property values for the asset in the form `Array(PnIDProperty1, … PnIDPropertyN)`. The names of the properties are returned by the AMMPropertiesToBeRequested Function.
AccumulatedData	A variant that contains user-defined data that can be accumulated from call to call and then passed to the AMMGetReportCustomHeaderValues Function.

Status parameter byte values

Value	Status	Vault Object Exists	Drawing Object Exists	Reference Exists	Tag Type of Vault Object is Matched with One of Drawing Object	Vault Objects with Duplicate Tag Values Exist	Drawing Objects with Duplicate Tag Values Exist	Comment
0	NotDefined	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available	Status is not calculated yet. This status cannot be sent during a correct request.
1	Ok	Yes	Yes	Yes	Yes	No	No	
2	NotLinked	Yes	Yes	No	Yes	No	No	
3	OnlyInDocument	No	Yes	Not Available	Not Available	Not Available	No	
4	MismatchFound	Yes	Yes	No	No	No	No	
5	MismatchReferenced	Yes	Yes	Yes	No	No	No	
6	CoercedReference	Yes	No	Yes	Not Available	No	Not Available	

Value	Status	Vault Object Exists	Drawing Object Exists	Reference Exists	Tag Type of Vault Object is Matched with One of Drawing Object	Vault Objects with Duplicate Tag Values Exist	Drawing Objects with Duplicate Tag Values Exist	Comment
7	OnlyInVault	Yes	No	No	Not Available	No	Not Available	Object with this status exists just after unlink vault object with CoercedReference status. If the object will not be relinked (by **Link** or **Assign**) then the row will be deleted on the next refresh.
8	DuplicatesInVault	Yes	condition is not checked	condition is not checked	condition is not checked	Yes	No	
9	DuplicatesInDrawing	condition is not checked	Yes	condition is not checked	condition is not checked	No	Yes	
10	Duplicates	Yes	Yes	condition is not checked	condition is not checked	Yes	Yes	

Value	Status	Vault Object Exists	Drawing Object Exists	Reference Exists	Tag Type of Vault Object is Matched with One of Drawing Object	Vault Objects with Duplicate Tag Values Exist	Drawing Objects with Duplicate Tag Values Exist	Comment
254	Error	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available	Status calculation failed for any reason.

Remarks

Called for every asset in the drawing.

AMMMainTagDocumentId Function

This user-defined function is invoked when users select a folder. Implement this function to return the ID of the main tag object associated with the active folder. If this function returns the ID of an existing document of the document type that is specified for the **Object document type** option on the **AMM Settings** tab, then the **Where Used** property page of the main tag object is shown for the active folder.

Syntax

```
Function AMMMainTagDocumentId ()
```

Parameters

This function accepts no parameters.

Example

The following example merely returns the first tag document found in the active folder. The function can be implemented with different search criteria to return a more specific tag document.

```
Function AMMMainTagDocumentId ()
    Dim Criteria
    Dim FolderDocuments

    Criteria = Array(Array _
                    'Internal name of the Parent Folder ID property

("AMDocumentPropertySet.*XC4800881716f11d1b47b000000000000%", _
                     IC_OP_EQUALS, Folder.ID))

    'Search in the folder
    Set FolderDocuments = Vault.FindDocuments(,Array("TagObject"),
Criteria, False)

    If FolderDocuments.Count > 0 Then
        'Return the first document ID
        'Your implementation should return the document ID
        'that meets your requirements
       AMMMainTagDocumentId = FolderDocuments.Document(0).ID
```

```
    End If
End Function
```

AMMPropertiesToBeRequested Function

Returns an array of asset properties values that will be passed to **AMMGetCustomColumnValues**.

Syntax

```
Function AMMPropertiesToBeRequested() As String
```

Parameters

This function accepts no parameters.

Example

```
Function AMMPropertiesToBeRequested()
    Dim mpropertyNames(1)
    mpropertyNames(0) = "_PREDEF_NAME_FOR_TYPE_DISP_NAME_"
    AMMPropertiesToBeRequested = mpropertyNames
End Function
```

Remarks

To retrieve the localized value of the **ClassName** property of AutoCAD P&ID tags, use the special property name **_PREDEF_NAME_FOR_TYPE_DISP_NAME_**.

AMMTags4TagPageIsVisible Function

Implement this user-defined function to control the visibility of the **Tags** property page for proxy tags in the Meridian Enterprise client applications.

Syntax

```
Function AMMTags4TagPageIsVisible ()
```

Parameters

This function accepts no parameters.

Example

```
Function AMMTags4TagPageIsVisible ()
    AMMTags4TagPageIsVisible = InStr(1, Document.FileName, "object")
= 1
End Function
```

AMMTagsManageLinksIsAllowed Function

User-defined function that controls activation of the **Manage Links** button in the **Manage Document Links** dialog box. The button is enabled if this function is not defined or returns **True**. The button is disabled if this function is defined and returns **False**.

Syntax

```
Function AMMTagsManageLinksIsAllowed()
```

Parameters

This function accepts no parameters.

Example

```
Function AMMTagsManageLinksIsAllowed(TagObjectID)
    ' Disable button for released tag
    Dim TagObject
    Set TagObject= Vault.GetDocument(TagObjectID)
       AMMTagsManageLinksIsAllowed = TagObject.WorkFlowState <> AS_WF_
RELEASED
End Function
```

Remarks

If the vault uses proxy documents for folder linking, the function must implement logic that determines whether the current document is a proxy document or is a main tag document. For more information about folder linking, see *Create And Link Proxy Documents* in the *Meridian Enterprise User's Guide*.

AMMTagsPageIsVisible Function

Implement this user-defined function to control the visibility of the **Tags** property page for documents in the Meridian Enterprise client applications. The page is visible if this function returns **True**.

Syntax

```
Function AMMTagsPageIsVisible ()
```

Parameters

This function accepts no parameters.

Example

```
Function AMMTagsPageIsVisible ()
    AMMTagsPageIsVisible = InStr(1, Document.FileName, "DWG") = 1
End Function
```

AMMUseMainTag Function

Implement this user-defined function to show the **Where Used** property page for folders in addition to documents in the Meridian Enterprise client applications. The folders must have an associated main tag. The tag is identified by the **AMMMainTagDocumentId** function described in AMMMainTagDocumentId Function. The page is visible if this function returns **True**.

Syntax

```
Function AMMUseMainTag()
```

Parameters

This function accepts no parameters.

Example

```
Function AMMUseMainTag()
    AMMUseMainTag = True
End Function
```

AMMWhereUsedManageLinksIsAllowed Function

User-defined function that controls activation of the **Manage Links** button in the **Manage Tag Links** dialog box. The button is enabled if this function is not defined or returns **True**. The button is disabled if this function is defined and returns **False**.

Syntax

```
Function AMMWhereUsedManageLinksIsAllowed()
```

Parameters

This function accepts no parameters.

Example

```
Function AMMWhereUsedManageLinksIsAllowed()
  ' The current document is a main tag
  ' Search for proxy document
  Dim Drawing
  Set Drawing = FindDoc(Document.ParentFolder.ID, "_M", ".dwg",
"AutoPlant")
  ' If proxy document was not found, use main tag.
  If Drawing Is Nothing Then
    Drawing = Document
  End If
  ' Allow editing when document is not released.
  AMMWhereUsedManageLinksIsAllowed = Drawing.WorkFlowState <> AS_WF_
RELEASED
End Function
```

Remarks

If the vault uses proxy documents for folder linking, the function must implement logic that determines whether the current document is a proxy document or a main tag document as shown in the preceding example. For more information about folder linking, see *Create And Link Proxy Documents* in the *Meridian Enterprise User's Guide*.

AMMWhereUsedPageIsVisible Function

Implement this user-defined function to control the visibility of the **Where Used** property page for proxy tags in the Meridian Enterprise client applications. The page is visible if this function returns **True**.

Syntax

```
Function AMMWhereUsedPageIsVisible()
```

Parameters

This function accepts no parameters.

Example

```
Function AMMWhereUsedPageIsVisible()
    AMMWhereUsedPageIsVisible = InStr(1, Document.FileName, "DWG") =
1
End Function
```

DebugAssert Function

During VBScript debugging, displays a message if the specified condition fails.

Syntax

```
DebugAssert(Condition As Boolean, FalseMessage As String)
```

Parameters

Name	Description
Condition	A conditional expression to evaluate.
FalseMessage	The message to display.

Remarks

Not available in PowerWeb.

FileExtension Function

Returns the file extension of a specified file.

Syntax

```
FileExtension(FileName As String) As String
```

Parameters

Name	Description
FileName	The full name of the file for which to return the file extension.

Remarks

Also returns the period, for example, .doc.

FileRoot Function

Returns the name of the specified file without the extension or path.

Syntax

```
FileRoot(FullFileName As String) As String
```

Parameters

Name	Description
FullFileName	The fully qualified path of a file for which to return the file root.

FormatSequenceAlfa Function

Returns a number converted into a character-based code and optionally prefixes characters until the specified string length is reached.

Syntax

```
FormatSequenceAlfa(Value As Long, [PrefixChar As String], [Length As Long = 0], [Compatibility As Boolean = False]) As String
```

Parameters

Name	Description
Value	The number to convert.
PrefixChar	The character to repeat as a prefix.
Length	The total length of the string to create.
Compatibility	If set to **True**, calculates a code that is compatible with the now obsolete AM-WorkFlow system.

Remarks

Use this function to format sequence numbers to a standard-length string for use in calculating document or folder names.

FormatSequenceAlfaNum Function

Returns a number converted into an alphanumeric code and optionally prefixes zeros until the specified string length is reached.

Syntax

```
FormatSequenceAlfaNum(Value As Long, [Length As Long = 0],
[Compatibility As Boolean = False]) As String
```

Parameters

Name	Description
Value	The number to convert.
Length	The total length of the string to create.
Compatibility	If set to **True**, calculates a code that is compatible with the now obsolete AM-WorkFlow system.

Remarks

Use this function to format sequence numbers to a standard-length string for use in calculating document or folder names.

FormatSequenceNum Function

Returns a number converted into a string and optionally prefixes zeros until the specified string length is reached.

Syntax

```
FormatSequenceNum(Value As Long, [Length As Long = 0]) As String
```

Parameters

Name	Description
Value	The number to convert.
Length	The total length of the string to create.

Remarks

Use this function to format sequence numbers to a standard-length string for use in calculating document or folder names.

GMTTime2Local Function

Converts a date from Greenwich Mean Time (GMT) to the time zone of the local computer.

Syntax

```
GMTTime2Local(GMTTime As Date) As Date
```

Parameters

Name	Description
GMTTime	A date expressed in GMT.

ListFromColumn Function

Returns a one-dimensional array of table column data.

Syntax

```
ListFromColumn(Table, [ColumnIndex As Long = 0])
```

Parameters

Name	Description
Table	An object that represents the Meridian table from which to return data.
ColumnIndex	Optional index number of the column from which to return data.

LocalTime2GMT Function

Converts a date from the local time zone to Greenwich Mean Time (GMT).

Syntax

```
LocalTime2GMT(LocalTime As Date) As Date
```

Parameters

Name	Description
LocalTime	A date expressed in the local time zone of the computer.

PnIDLink_GetAssetCoordinates Function

Recalculates the asset coordinate to another format.

Syntax

```
Function PnIDLink_GetAssetCoordinates (X As String, Y As String,
SheetSize and String) As String
```

Parameters

Name	Description
X	Absolute coordinate X (in drawing units) of the asset in the drawing.
Y	Absolute coordinate Y (in drawing units) of the asset in the drawing.
SheetSize	The value returned by the PnIDLink_GetSheetSize Function.

Return Value

Returns array of strings (X,Y).

Remarks

Called for every asset in the drawing before **AMMGetCustomColumnValues**.

PnIDLink_GetSheetSize Function

Returns the name of the sheet size for the current drawing.

Syntax

```
Function PnIDLink_GetSheetSize (BlockList As String) As String
```

Parameters

Name	Description
BlockList	An array of titleblock names in the drawing.

Remarks

This function is called once before **PnIDLink_GetAssetCoordinates**.

PnIDLink_IsMainDrawing Function

Returns a Boolean value indicating whether the Meridian functionality is enabled for the current drawing.

Syntax

```
Function PnIDLink_IsMainDrawing (Args As String) As Boolean
```

Parameters

Name	Description
Args	The local workspace or shared workspace path of the drawing.

Remarks

Called once when the drawing is opened in AutoCAD P&ID.

Quote Function

Returns a string enclosed in a specified quotation character.

Syntax

```
Quote(Value As String,[QuoteChar As String]) As String
```

Parameters

Name	Description
Value	The string to enclose.
QuoteChar	The character to enclose the string.

ValidateFolderName Function

Replaces illegal characters in a file or folder name with a specified character.

Syntax

```
ValidateFolderName(Name As String, ReplacementChar As String) As String
```

Parameters

Name	Description
Name	The file or folder name to validate.
ReplacementChar	The character with which to replace illegal characters.

Remarks

Use this function to validate newly calculated file or folder names.

WinInputBox Function

Shows a modal dialog box to receive input from the user.

Syntax

```
WinInputBox (Prompt As String, [Title As String], [Default As
String]) As String
```

Parameters

Name	Description
Prompt	Body text of the dialog.
Title	Title bar text of the dialog.
Default	Default input text of the dialog.

Remarks

Not available in PowerWeb. The **Prompt** text can be formatted, as described in Formatting Text With RTF Codes. To detect if the user clicked the **Cancel** button, test the returned value for an empty string ("").

WinMsgBox Function

Shows a configurable dialog box containing a message to the user.

Syntax

```
WinMsgBox (Prompt As String, [Style As AS_MsgBoxStyle = AS_
ApplicationModal], [Title As String]) As AS_MsgBoxResult
```

Parameters

Name	Description
Prompt	Body text of the dialog.
Style	Style of the dialog as one or more of the **AS_MsgBoxStyle** constants.
Title	Title bar text of the dialog.

Remarks

Not available in PowerWeb. For PowerWeb, you can use Confirmation pages. See our KnowledgeBase article to learn more about how Confirmation Pages can be implemented.

The **Prompt** text can be formatted as described in Formatting Text With RTF Codes.

Creating Custom Functions

The Meridian implementation of VBScript supports custom functions. These functions can be used the same as Visual Basic functions. This means you can define functions in the Meridian**Events** code block and use them either in event procedures or in configuration expressions. Your custom functions appear in the **(General)** branch of the **Events and Procedures** list when editing the **Events** code block and in the **(General)** branch of the **Object Browser** when editing configuration expressions. This makes it easy to write a function once and to reuse it in multiple places. For more information about creating and using custom functions, see the VBScript Language Reference web site.

> **Note:**
> Custom functions are not available in the **Database Import Wizard**.

Meridian Event Procedures

Meridian event procedures allow you to customize the way Meridian works by adding your own functionality onto what Meridian already does. When a user executes certain commands in one of the Meridian client applications, Meridian performs a sequence of actions. At various steps in the sequence, Meridian declares that an action (event) is occurring to which VBScript can react. Meridian checks for the definition of a specific VBScript procedure (event handler) that corresponds to each event. If the procedure exists, Meridian executes it before proceeding on to the next step in the sequence. All VBScript event procedures are stored in a code block in the vault separate from the code blocks for the configuration expressions.

Event procedures are more powerful than configuration expressions. Each configuration expression is a single statement to be evaluated. Event procedures can contain many statements and provide additional advantages. Event procedures can:

- Use variables
- Use subroutines
- Change the values of properties
- Invoke the methods of objects, for example, **Document.RevokeWorkflow**

> **Note:**
> The order of the events may change slightly in new releases to accommodate new functionality or to resolve problems and cannot be guaranteed. Test all customization that relies on event procedures with each upgrade before deploying the customization for production use.

Names Of Events

The names and syntax of event procedures are predefined. You can edit only the body. A prototype of a procedure looks like the following example:

```
Sub DocGenericEvent_AfterPrint(Batch)
            'Add your code here
End Sub
```

The names of event procedures include the event category and the event name. In the example above, the event category is **DocGenericEvent** and the event name is **AfterPrint**. With event procedure names standardized this way, you can easily identify the purpose of each procedure. You can view all of the Meridian event procedures with the **Events and Procedures** browser in the Meridian Enterprise Script Editor.

> **Note:**
> For brevity and readability, some of the event names in this guide show an asterisk * in the position of the procedure name that represents the event type. This means that there are separate **Initialize**, **Before**, **After**, and **Terminate** event procedures for the event, but they are all described in the same topic. For example, following are the event procedure names for the **Print** event:

- DocGenericEvent_InitializePrint
- DocGenericEvent_BeforePrint
- DocGenericEvent_AfterPrint
- DocGenericEvent_TerminatePrint

In this example, all four event procedures listed above are described in the **DocGenericEvent_ *Print** section.

Batch Events

All document events are batch events. The batch size depends on the number of documents the user has selected when the event occurs. A single-document operation is a batch operation with size 1. For example, if the user selects four documents and then uses the **Start Change** command, the event procedures will execute on a batch of four documents. If the user has selected only a single document, only that document is processed, but all batch events still occur in order.

Order Of Events

Most of the event procedures consist of four types for each event:

- **Initialize**: Your code should use these types of event procedures to prepare for batch operations
- **Before** and **After**: Your code should perform the following:
    1. Check conditions for the individual documents
    2. Change property values for the individual documents
    3. Perform necessary actions on the individual documents
- **Terminate**: Your code should invoke any necessary user interface procedures and release resources. This is typically some overview of the result.

Within a single event, the event procedures are invoked in the above order. The **Initialize** and **Terminate** procedures are invoked once each for the entire batch and for most of the events only the **Vault** object is available. The **Before** and **After** procedures are invoked once for each object (document, folder, and so on) affected by the event and during these events, the affected objects are available.

Your code should not assume that all of the event types will occur under all conditions. For example, if an unexpected error occurs during an operation, the **After** or **Terminate** events might not occur.

The command a user executes may also involve more than one type of event. The following example lists the **NewDocument** and **CalculateFileName** events and the order in which they occur when a new document is created:

1. **DocGenericEvent_InitializeNewDocument**
2. **DocGenericEvent_InitializeCalculateFileName**
3. **DocGenericEvent_BeforeNewDocument**
4. **DocGenericEvent_AfterNewDocument**
5. **DocGenericEvent_BeforeCalculateFileName**
6. **DocGenericEvent_AfterCalculateFileName**
7. **DocGenericEvent_TerminateCalculateFileName**
8. **DocGenericEvent_TerminateNewDocument**

The preceding example is a relatively simple one in which all of the events are in the **DocGenericEvent** category and only one document is involved. In the following example, events in the **DocGenericEvent** and **DocCopyMoveEvent** categories occur in the order listed when a batch of documents (some of which might have references) is copied:

1. Before the batch of documents is processed:
    a. **DocCopyMoveEvent_PrepareCopy**
    b. **DocCopyMoveEvent_InitializeCopy**
    c. **DocGenericEvent_InitializeNewDocument**
    d. **DocGenericEvent_InitializeCalculateFileName**
2. For each source document in the batch:
    a. **DocCopyMoveEvent_BeforeCopyWithReferences** (if the source document has references)
    b. **DocCopyMoveEvent_BeforeCopy**
    c. **DocCopyMoveEvent_AfterCopy**
    d. **DocCopyMoveEvent_BeforeCopy** (for each reference of the current source document)
    e. **DocCopyMoveEvent_AfterCopy** (for each reference of the current source document)
    f. **DocCopyMoveEvent_AfterCopyWithReferences** (if the source document has references)
3. For each destination document (copy made) in the batch:
    a. **DocGenericEvent_BeforeNewDocument**
    b. **DocGenericEvent_AfterNewDocument**
    c. **DocGenericEvent_BeforeCalculateFileName**
    d. **DocGenericEvent_AfterCalculateFileName**
4. After the batch of documents has been processed:
    a. **DocGenericEvent_TerminateCalculateFileName**
    b. **DocGenericEvent_TerminateNewDocument**
    c. **DocCopyMoveEvent_TerminateCopy**

These same events occur when documents are moved, derived, or replaced but with the corresponding events (**DocCopyMoveEvent_InitializeMove**, **DocCopyMoveEvent_BeforeMove**, and so on) instead of **DocCopyMoveEvent_InitializeCopy**, **DocCopyMoveEvent_BeforeCopy**, and so on. The only difference is that the source and destination documents are the same document.

The order of the events within each category for specific Meridian commands are listed in the concept topic for each event category, Document Generic Events, for example.

Create and Edit Event Procedures

By default, a new vault has no custom event procedures. Only the built-in Meridian functionality will execute when events occur. All custom procedures (other than event procedures) and functions are listed in the **General** category and do not react directly to events. If you write a procedure or function that is to be called from within one of the event procedures, it will be listed in the **General** category.

Many of the parameters of the Meridian event procedures are objects. You can view all of the Meridian objects and their methods and properties with the **Object Browser** of the Meridian Enterprise Script Editor. Some of the Meridian event procedures accept as parameters one or more predefined Meridian constants. The available Meridian constants can also be found in the **Object Browser** in the Meridian Enterprise Script Editor.

To create or edit an event procedure:

1. In Configurator, on the **Edit** menu, select **Edit Events**.

   The Meridian Enterprise Script Editor appears showing the existing custom event procedures, if any.

2. From the **Events and Procedures** list, select the event that you want to create or edit.

   If a custom event procedure already exists, the code pane is scrolled to show the procedure's definition and the mouse cursor is moved to the first line of the procedure. If a custom event procedure does not already exist, a prototype of the definition is added to the end of the existing code and the mouse cursor is moved to the first line of the procedure.

3. Add or edit code for the procedure using the **Object Browser** and **Events and Procedures** list as necessary.

4. Choose between two options:

   - Click **OK** to keep your changes when you are finished editing.
   - Click **Cancel** to abandon your changes.

5. On the **Vault** menu, select **Save** to save your changes to the vault where they are available to users the next time they open the vault.

Limiting Events Generated By VBScript

The code that is executed in one event procedure can cause another event. In other words, one event can cause a chain reaction of other events that can make it difficult to limit the effects of the original event procedure. For example, the **Document.MoveTo** method can invoke the **NewDocument** events when the document is created in the destination folder. You might not want that to happen.

You can limit which events are generated by certain event procedures and avoid an event procedure intended for one purpose from being executed by actions for which it was not intended.

To limit the events generated by event procedures:

1. In Configurator, expand **Environment** and select **Application Link Settings**.

   The settings for each link appear in property pages in the right pane.
2. Click **Application Integration**.

   The **Application Integration** settings page appears in the right pane.
3. Click **Edit**.
4. Locate the line that contains **[ScriptEvents]**.

   This section of the configuration controls the events that are generated by VBScript event procedures. A number of settings are listed in this section that each control a group of events. Each setting is described in the comments that precede it.
5. Read the comments for each setting and change the setting to **Y** (Yes) or **N** (No) to meet your needs.
6. Click **OK**.

Asset Management Events

Asset management events occur when users work with the commands provided by the Meridian Enterprise Asset Management module.

AIMS_AddComment Event

Occurs after a user has added a comment to a document in Meridian Explorer.

Syntax

```
Function AIMS_AddComment (commentText As String, attachmentType As String, numberOfComments As Long)
```

Parameters

Name	Description
commentText	The text of the comment.
attachmentType	The type of file attached to the comment. The possible values are: **0**: No attachment **1**: Redline **2**: Image
numberOfComments	The total number of comments on the document.

Example

```
Function AIMS_AddComment(commentText, attachmentType,
numberOfComments)
    Document.Log "A comment was added to " + Document.FileName +
vbNewLine + "Text: " + commentText
   Select Case attachmentType
       Case "Redline"
           Document.Log "A redline markup was added to " +
Document.FileName
       Case "File"
           Document.Log "A file was attached to " + Document.FileName
```

```
   End Select
End Function
```

AIMS_Attach_Intialize Event

Occurs after a user has invoked the **Upload** command in Meridian Explorer to upload a file but before the document is created in the vault. This event can be used to get the source filename and the URL parameters of the Meridian Explorer page. Its return values can be used to allow the upload to proceed and to specify the destination folder and document type or to cancel the operation.

This event is required for the **Upload** command. The command must be enabled as described in *Compose View URLs* in the *Meridian Enterprise Server Administrator's Guide*.

Syntax

```
Function AIMS_Attach_Intialize (SourceFilename As String, QueryString
As Array) As Array
```

Parameters

Name	Description
SourceFilename	The original filename that is uploaded from the client.
QueryString	An array of name-value pairs for the URL parameters of the Meridian Explorer page.

Returned array

Value	Description
Result	Boolean **True** if the event should succeed, **False** if it should fail. If the event should fail, the upload is aborted and an error message should be returned as the second item in the array and the following items should not be returned.
TargetPath	The path of the target folder in the vault where the uploaded file should be stored.
DocumentType	The internal name of the document type that should be assigned to the new document.

Example

```
' Event handler for BC-Explorer Asset Management Module link
' Returns required input for the Upload function
```

```
Function AIMS_Attach_Intialize(sourceFileName, queryString)
    Dim targetFolder
    Dim docTypeName

    ' Determine the target folder for the uploaded document
    Set targetFolder = Vault.RootFolder.GetSubFolder("Miscellaneous")
    If Not User.HasPrivilege(AS_PRIVILEGE_DOCUMENT_CREATE,
targetFolder) Then
        ' Abort the upload
        AIMS_Attach_Intialize = Array(False, "Upload is not allowed")
        Exit Function
    End If

    ' Determine the document type for the uploaded folder
    docTypeName = "GenericDocument"
    If (LCase(Right(sourceFileName, 4)) = ".dwg") Then
        docTypeName = "Drawing"
    End If

    ' Return an array with the target folder path and the document
type name
    AIMS_Attach_Intialize = Array(True, targetFolder.Path,
docTypeName)
End Function
```

AIMS_Attach_Before Event

Occurs after a user has invoked the **Upload** command in Meridian Explorer to upload a file and the document object has been created in the vault. It occurs before the document content has been imported, its properties have been set, and the reference created. This event can be used to set the properties and workflow state of the **Document** object. Its return values can be used to allow the operation to proceed or to cancel the operation. This event is optional for the **Upload** command.

Syntax

```
Function AIMS_Attach_Before (SourceFilename As String, QueryString As
Array) As Array
```

Parameters

Name	Description
SourceFilename	The original filename that is uploaded from the client.

Name	Description
QueryString	An array of name-value pairs for the URL parameters of the Meridian Explorer page.

Returned array

Value	Description
Result	Boolean **True** if the event should succeed, **False** if it should fail. If the event should fail, the document creation is aborted and an error message should be returned as the second item in the array.
Message	An error message to show the user why the operation has been aborted.

Example

```
' Event handler for BC-Explorer Asset Management Module link
' Invoked before the uploaded content is imported to the document
Function AIMS_Attach_Before(sourceFileName, queryString)
    Dim viewID
    Dim tagFilter
    Dim tagnr

    ' You may retrieve values from the URL of the BC-Explorer Related
Documents page
    viewID = AIMS_GetQueryStringValue(queryString, "VIEWID")
    tagFilter = AIMS_GetQueryStringValue(queryString, "TAGFILTER")
    tagnr = AIMS_GetQueryStringValue(queryString, "TAGNR")

    If False Then
        ' You may abort the upload if required
        AIMS_Attach_Before = Array(False, "Some reason to stop the
upload")
        Exit Function
    End If

    AIMS_Attach_Before = Array(True, "")
End Function


' Extract the value from the name-value collection
Function AIMS_GetQueryStringValue(queryString, name
    Dim index

    For index = LBound(queryString, 1) To UBound(queryString, 1)
       ' Find the named value
```

```
        If (UCase(Cstr(QueryString(index)(0))) = UCase(CStr(name)))
Then
                ' Return the value
                AIMS_GetQueryStringValue = Cstr(QueryString(index)(1))
                Exit Function
        End If
    Next

    AIMS_GetQueryStringValue = ""
End Function
```

Remarks

The first source vault of the repository is used as the destination. If a document with the same name already exists in the vault, a number will be appended to the name, for example, **MyFile(1)**.

AIMS_Attach_After Event

Occurs after a user has invoked the **Upload** command in Meridian Explorer and:

- The document tag has been created in the vault
- Its content has been imported
- Its properties have been set
- The reference to the Meridian Explorer item as been created
- But *before* the transaction has been committed.

This event can be used to modify the **Document** object and to allow the operation to complete or to cancel the operation. This event is optional for the **Upload** command.

Syntax

```
Function AIMS_Attach_After (SourceFilename As String, QueryString As
Array) As Array
```

Parameters

Name	Description
SourceFilename	The original filename that is uploaded from the client.
QueryString	An array of name-value pairs for the URL parameters of the Meridian Explorer page.

Returned array

Value	Description
Result	Boolean **True** if the event should succeed, **False** if it should fail. If the event should fail, the document creation is aborted and an error message should be returned as the second item in the array.
Message	An error message to show the user why the operation has been aborted.

Example

```
' Event handler for BC-Explorer Asset Management Module link
' Invoked after the uploaded content is imported to the document
Function AIMS_Attach_After(sourceFileName, queryString)
    Dim viewID
    Dim tagFilter
    Dim tagnr

    ' You may retrieve values from the URL of the BC-Explorer Related
Documents page
    viewID = AIMS_GetQueryStringValue(queryString, "VIEWID")
    tagFilter = AIMS_GetQueryStringValue(queryString, "TAGFILTER")
    tagnr = AIMS_GetQueryStringValue(queryString, "TAGNR")

    ' Create a reference to the asset
    Dim doc
    Dim criteria
    criteria = Array(Array("TagInfo.TagNr", IC_OP_EQUALS, tagnr))
    For Each doc In Vault.FindDocuments(, Array("TagObject"),
criteria, False)
        Document.GetReferences("TagObjectReference",False).Add
(doc.ID)
    Next

    If False Then
        ' You may abort the upload if required
        AIMS_Attach_After = Array(False, "Some reason to stop the
upload")
        Exit Function
    End If

    ' Release the uploaded document
    Call Document.ChangeWorkflowState(AS_WF_RELEASED, "", User.Name)
    Document.Log User.Name + " uploaded '" + sourceFileName + "' to "
+ Document.Path

    ' Return a user message
```

```
    AIMS_Attach_After = Array(True, "Upload completed")
End Function

' Extract the value from the name-value collection
Function AIMS_GetQueryStringValue(queryString, name
    Dim index

    For index = LBound(queryString, 1) To UBound(queryString, 1)
       ' Find the named value
        If (UCase(Cstr(QueryString(index)(0))) = UCase(CStr(name)))
Then
               ' Return the value
               AIMS_GetQueryStringValue = Cstr(QueryString(index)(1))
               Exit Function
        End If
    Next

    AIMS_GetQueryStringValue = ""
End Function
```

AIMS_DeleteComment Event

Occurs after a user has deleted a comment from a document in Meridian Explorer.

Syntax

```
Function AIMS_DeleteComment (commentText As String, attachmentType As
String, numberOfComments As Long)
```

Parameters

Name	Description
commentText	The text of the comment.
attachmentType	The type of file attached to the comment. The possible values are: **0**: No attachment **1**: Redline **2**: Image
numberOfComments	The total number of comments on the document.

Example

```
Function AIMS_DeleteComment(commentText, attachmentType,
numberOfComments)
   If numberOfComments = 0 Then
       Document.Log "All comments have been removed from " +
Document.FileName + vbNewLine + "Text: " + commentText
   Else
       Document.Log "A comment was removed from " + Document.FileName
+ vbNewLine + "Text: " + commentText
   End If
End Function
```

AIMS_CloseComment Event

Occurs after a user has closed a discussion on a document in Meridian Explorer.

Syntax

```
Function AIMS_CloseComment (commentText As String, attachmentType As
String, numberOfComments As Long)
```

Parameters

Name	Description
commentText	The text of the comment.
attachmentType	The type of file attached to the comment. The possible values are: **0**: No attachment **1**: Redline **2**: Image
numberOfComments	The total number of comments on the document.

Example

```
Function AIMS_CloseComment(commentText, attachmentType,
numberOfComments)
    Document.Log "A comment was closed for " + Document.FileName +
vbNewLine + "Text: " + commentText
End Function
```

AIMS_UpdateComment Event

Occurs after a user has updated a comment on a document in Meridian Explorer.

Syntax

```
Function AIMS_UpdateComment (commentText As String, attachmentType As
String, numberOfComments As Long)
```

Parameters

Name	Description
commentText	The text of the comment.
attachmentType	The type of file attached to the comment. The possible values are: **0**: No attachment **1**: Redline **2**: Image
numberOfComments	The total number of comments on the document.

Example

```
Function AIMS_UpdateComment(commentText, attachmentType,
numberOfComments)
   Document.Log "A comment was updated for " + Document.FileName +
vbNewLine + "Text: " + commentText

   Select Case attachmentType
       Case "Redline"
           Document.Log "A redline markup was added to " +
Document.FileName
       Case "File"
           Document.Log "A file was attached to " + Document.FileName

   End Select
End Function
```

ObjectsPage_IsVisible Event

Occurs before the **Objects** HTML page is shown in PowerWeb.

Syntax

```
ObjectsPage_IsVisible As Boolean
```

Remarks

The **Objects** HTML page is an alternative implementation of the default **Objects** page that is shown in PowerUser. The HTML page has equivalent functionality that is not available in the default page shown by the **AssetManagementTags** extension in PowerWeb. The HTML page is only for use in PowerWeb and the **AssetManagementTags** extension should not be assigned to any document types.

Return **False** (default) to hide the property page.

Example

```
Function ObjectsPage_IsVisible()
  ObjectsPage_IsVisible = True
End Function
```

WhereUsedPage_IsVisible Event

Occurs before the **Where Used** HTML page is shown in PowerWeb.

Syntax

```
WhereUsedPage_IsVisible As Boolean
```

Remarks

The **Where Used** HTML page is an alternative implementation of the default **Where Used** page that is shown in PowerUser. The HTML page has equivalent functionality that is not available in the default page shown by the **AssetManagementWhereUsed** extension in PowerWeb. The HTML page is only for use in PowerWeb and the **AssetManagementWhereUsed** extension should not be assigned to any document types.

Return **False** (default) to hide the property page.

Example

```
Function WhereUsedPage_IsVisible()
  WhereUsedPage_IsVisible = True
End Function
```

CAD Link Events

CAD link events allow you to customize the behavior of the links to the various applications that Meridian supports. These events occur when the link commands are run from within the Meridian client applications and when the link commands are run from within the linked applications.

CAD Link Event Sequences

The events that occur for the application link commands are shown in the following lists in the sequence that they occur.

Synchronize properties from file event sequence

1. **DocCADLink_AfterReadMTBProperties**
2. **DocGenericEvent_PrepareCommand**
3. **DocCADLink_InitializeUpdateProperties**
4. **DocCADLink_BeforeUpdateProperties**
5. **DocCADLink_AfterReadProperty**: Once for each property.
6. **DocCADLink_AfterUpdateProperties**
7. **DocCADLink_TerminateUpdateProperties**

Synchronize properties to file event sequence

1. **DocCADLink_BeforeWriteMTBProperties**
2. **DocCADLink_InitializeUpdateProperties**
3. **DocCADLink_BeforeUpdateProperties**
4. **DocCADLink_BeforeWriteProperty**: Once for each property.
5. **DocCADLink_AfterUpdateProperties**
6. **DocCADLink_TerminateUpdateProperties**

Synchronize references from file event sequence

1. **DocCADLink_InitializeUpdateReferences**
2. **DocCADLink_BeforeUpdateReferences**
3. **DocCADLink_AfterUpdateReferences**
4. **DocCADLink_TerminateUpdateReferences**

AfterReadProperty Event

Occurs after a document property that is mapped to a title block attribute is read.

Syntax

```
DocCADLink_AfterReadProperty(Batch, PropertyName, Value)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
PropertyName	A string that contains the property name that was read.
Value	The property value that was read.

Remarks

This event also occurs after a title block is read during batch import by the Document Import tool. The **Document** object is available during this event but is read-only.

AfterReadMTBProperties Event

Occurs after a document property that is mapped to multiple title block attributes is read.

Syntax

```
DocCADLink_AfterReadMTBProperties (Batch, BCPropStorage)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
BCPropStorage	An object that contains the property information that will be written. See BCPropStorage Object for its available properties and methods.

Example

```
Sub DocCADLink_AfterReadMTBProperties(Batch, BCPropStorage)
  Dim rowCollection
  Dim allColDefs
  Dim colDef
  Dim row
  Set allColDefs = BCPropStorage.GetColumnDefs(2)
  Set rowCollection = BCPropStorage.Rows

  For Each row in rowCollection
    For Each colDef In allColDefs
      If Not IsNull(row.Property(colDef.Name).Value) Then
        WinMsgBox "AfterReadMTB_Layout: " & _
          row.Property( "Layout").Value & vbCrLf & _
          "Property: " & CStr(colDef.Name) & vbCrLf & _
          "Value: " & CStr(row.Property(colDef.Name).Value)
      End If
    Next
  Next
End Sub
```

BeforeWriteProperty Event

Occurs before a document property that is mapped to a title block attribute is written.

Syntax

```
DocCADLink_BeforeWriteProperty(Batch, PropertyName, Value)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Name	Description
PropertyName	A string that contains the property name that will be written to.
Value	The property value that will be written.

Remarks

The **Document** object is available during this event but is read-only.

BeforeWriteMTBProperties Event

Occurs before a document property that is mapped to multiple title block attributes is written.

Syntax

```
DocCADLink_BeforeWriteMTBProperties (Batch, BCPropStorage)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
BCPropStorage	An object that contains the property information that will be written. See BCPropStorage Object for its available properties and methods.

Example

```
Sub DocCADLink_BeforeWriteMTBProperties(Batch, MTBProperties)
  Document.Log "BeforeWriteMTBProperties -->"
  Dim rowCollection, allColDefs, colDef, row

  Set allColDefs = MTBProperties.GetColumnDefs(2)
  Set rowCollection = MTBProperties.Rows

  For Each row in rowCollection
    For Each colDef In allColDefs
      If Not row.Property(colDef.Name) Is Nothing Then
        Dim rv: rv = "AfterReadMTB_Layout: " & _
          row.Property("Layout").Value & vbCrLf & _
          "Property: " & CStr(colDef.Name) & vbCrLf & _
          "Value: " & CStr(row.Property(colDef.Name).Value)
        Document.Log rv
```

```
      End If
    Next
  Next
End Sub
```

OnUpdateReference Event

Occurs when a specific reference to or from the selected documents is updated.

Syntax

`DocCADLink_OnUpdateReference(`*`Batch`*`, `*`RefFile`*`, `*`IsCreated`*`, `*`ToFile`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
RefFile	A string that contains the name of the file that is referenced.
IsCreated	**True** if the reference exists, **False** if the reference does not yet exist.
ToFile	Set to **True** to update the reference information stored in the file, **False** to only update the references in the vault.

*UpdateProperties Events

Occurs when document properties that are mapped to title block attributes are synchronized.

Syntax

`DocCADLink_*UpdateProperties(`*`Batch`*`, `*`ToFile`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
ToFile	Set to **True** to update the title block, **False** to only update the document properties.

*UpdateReferences Events

Occur when references to and from the document are updated.

Syntax

```
DocCADLink_*UpdateReferences(Batch, ToFile)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
ToFile	Set to **True** to update the reference information stored in the file, **False** to only update the references in the vault.

Custom Command Events

Custom command events occur when a custom command is executed.

Custom command event sequence

The following events occur when a document is selected.

1. **<CommandName>_State**

   The _State event determines what commands are available in the menu ribbon.
2. **DocGenericEvent_OnProperties**

The events that occur for custom commands are shown in the following list in the sequence that they occur.

1. **DocGenericEvent_PrepareCommand**
2. **<CommandName>_PreInitialize**
3. **<CommandName>_Initialize**
4. **<CommandName>_BeforeWizard**
5. **<CommandName>_AfterWizard**
6. **<CommandName>_PreExecute**
7. **<CommandName>_Execute**
8. **<CommandName>_Terminate**

<CommandName>_State Event

Occurs when the client application is calculating which custom commands should be available in the main menu, ribbon, or context menu.

Syntax

```
<CommandName>_State(Mode)
```

Parameters

Name	Description
Mode	Long integer that represents one or more **AS_CMD_MODE** constants.

Remarks

Use this event to set the status of the menu item to a long integer that represents one or more **AS_CMD_STATE** constants.

<CommandName>_Initialize Event

Occurs before the command is executed.

Syntax

```
<CommandName>_Initialize(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

<CommandName>_BeforeWizard Event

Occurs before a wizard page is shown.

Syntax

```
<CommandName>_BeforeWizard(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

<CommandName>_AfterWizard Event

Occurs after a wizard page is shown.

Syntax

```
<CommandName>_AfterWizard(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

<CommandName>_Execute Event

Occurs when the command is executed.

Syntax

```
<CommandName>_Execute(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Remarks

Use this event to execute the main logic of the command.

<CommandName>_PreExecute Event

Occurs before the command has been executed. This event is used to add a confirmation page to the command.

Syntax

```
<CommandName>_PreExecute(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Remarks

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

<CommandName>_PreInitialize Event

Occurs before the command is initialized. This event is used to add a confirmation page to the command.

Syntax

```
<CommandName>_PreInitialize(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Remarks

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

<CommandName>_Terminate Event

Occurs after the command has executed.

Syntax

```
<CommandName>_Terminate(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Custom Page Events

Custom page events allow you to control property pages you created in the Configurator. If you want to control the behavior of an out-of-the-box system property page, see Property Page Events.

Custom Page Event Sequence

The events that occur for custom pages are shown in the following list in the sequence that they occur.

- CustomPages_IsVisible
- CustomPages_CanEdit
- CustomPages_CanApply(Context)

CustomPages_IsVisible Event

Controls whether custom property pages are visible.

Syntax

```
CustomPages_IsVisible As Boolean
```

CustomPages_CanEdit Event

Controls whether the **Edit** button appears on custom property pages in the Meridian client applications. Set to **False** to prevent users from editing custom property pages.

This setting does not control out-of-the-box system property pages, such as the **Document** property page. If you want to disable editing for a system property page, implement the corresponding <PageName>_CanEdit() event.

Syntax

```
CustomPages_CanEdit As Boolean
```

CustomPages_CanApply Event

Controls whether data changes are accepted for editable custom property pages in the Meridian client applications. If the user enters changes but the **CustomPages_CanApply** event is set to **False**, Meridian will silently reject submitted changes.

Syntax

```
CustomPages_CanApply(Context) As Boolean
```

Parameters

Name	Description
Context	A one dimension array that contains the layout name, block name, property name, old value, and new value. Read only.

Document Copy/Move Events

Some events occur when documents are moved or created as copied, derived, or replacement documents.

Events which have an asterisk (*) in their name in the expanding sections below also have precursor events which occur prior to the main event. For example, the **PreBeforeCopy** event occurs before the **BeforeCopy** event. Both the precursor event and its subsequent event are described in the same expanding section.

Document Copy/Move Event Sequences

The events that occur for the document copy and move commands are shown in the following lists in the sequence that they occur.

Copy, Paste, Derive, Replace event sequence

1. **DocCopyMoveEvent_SelectTarget**
2. **DocCopyMoveEvent_PrepareCopy**: Source document only.
3. **DocCopyMoveEvent_PreInitializeCopy**
4. **DocCopyMoveEvent_InitializeCopy**: **Copy** and **Paste** commands only.
5. **DocCopyMoveEvent_InitializeDerive**: **Derive** command only.
6. **DocCopyMoveEvent_InitializeReplace**: **Replace** command only.
7. **DocGenericEvent_InitializeNewDocument**
8. **DocGenericEvent_InitializeCalculateFileName**
9. **DocCopyMoveEvent_PreBeforeCopyWithReferences**
10. **DocCopyMoveEvent_BeforeCopyWithReferences**: **Copy with References command** and **Derive with References** commands only.
11. **DocCopyMoveEvent_PreBeforeCopy**: PreBeforeCopy is not called when executing **Copy with References**. Copy with References will call PreBeforeCopyWithReferences.
12. **DocCopyMoveEvent_BeforeCopy**: **Copy** and **Paste** commands only.
13. **DocCopyMoveEvent_AfterCopy**: **Copy** and **Paste** commands only.
14. **DocCopyMoveEvent_BeforeDerive**: **Derive** command only.
15. **DocCopyMoveEvent_AfterDerive**: **Derive** command only.
16. **DocCopyMoveEvent_BeforeReplace**: **Replace** command only.

17. **DocCopyMoveEvent_AfterReplace**: **Replace** command only.
18. **DocGenericEvent_BeforeNewDocument**
19. **DocGenericEvent_OnProperties**: Except **Paste** command.
20. **DocGenericEvent_AfterNewDocument**
21. **DocCopyMoveEvent_AfterCopyWithReferences**: **Copy with References command** and **Derive with References** commands only.
22. **DocGenericEvent_TerminateCalculateFileName**
23. **DocGenericEvent_TerminateNewDocument**
24. **DocCopyMoveEvent_TerminateCopy**: **Copy** and **Paste** commands only.
25. **DocCopyMoveEvent_TerminateDerive**: **Derive** command only.
26. **DocCopyMoveEvent_TerminateReplace**: **Replace** command only.

Move event sequence

1. **DocCopyMoveEvent_InitializeMove**
2. **DocGenericEvent_InitializeNewDocument**
3. **DocGenericEvent_InitializeCalculateFileName**
4. **DocCopyMoveEvent_BeforeMove**
5. **DocCopyMoveEvent_AfterMove**
6. **DocGenericEvent_BeforeNewDocument**
7. **DocGenericEvent_OnProperties**
8. **DocGenericEvent_AfterNewDocument**
9. **DocGenericEvent_TerminateCalculateFileName**
10. **DocGenericEvent_TerminateNewDocument**
11. **DocCopyMoveEvent_TerminateMove**

AfterCopy Event

Occurs after documents are copied and pasted. Applies to **Copy** and **Paste** commands only.

Syntax

`DocCopyMoveEvent_AfterCopy(`*`Batch, TargetFolder`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

AfterCopyWithReferences Event

Occurs after documents are copied with references. Applies to **Copy with References command** and **Derive with References** commands only.

Syntax

```
DocCopyMoveEvent_AfterCopyWithReferences(Batch, SourceDocument,
OldParentDocument, NewParentDocument)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
SourceDocument	An object that represents the source document.
OldParentDocument	An object that represents the parent document of the source documents.
NewParentDocument	An object that represents the parent document of the copied documents.

AfterDerive Event

Occurs after documents are derived. Applies to **Derive** command only.

Syntax

```
DocCopyMoveEvent_AfterDerive(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

AfterMove Event

Occurs after documents are moved.

Syntax

```
DocCopyMoveEvent_AfterMove(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

AfterReplace Event

Occurs after documents are replaced.

Syntax

```
DocCopyMoveEvent_AfterReplace(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

*BeforeCopy Event

Occurs before documents are copied. The **PreBeforeCopy** event can be used to add a confirmation page before the **BeforeCopy** event. This event can be used as part of a wizard.

PreBeforeCopy is not called when executing the **Copy with References** command. Copy with References will call **PreBeforeCopyWithReferences**.

Syntax

```
DocCopyMoveEvent_*BeforeCopy(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

Remarks

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

*BeforeCopyWithReferences Event

Occurs before documents are copied with references. Applies to **Copy with References command** and **Derive with References** commands only.

The **PreBeforeCopyWithReferences** event occurs before the **BeforeCopyWithReferences** event. PreBeforeCopyWithReferences can be used to add a confirmation page before the **Copy with References Wizard**.

Syntax

```
DocCopyMoveEvent_*BeforeCopyWithReferences(Batch, OldParentDocument,
NewParentDocument, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
OldParentDocument	An object that represents the parent document of the source documents.
NewParentDocument	An object that represents the parent document of the copied documents.
TargetFolder	An object that represents the destination folder. This event can modify the destination folder.

Remarks

When a project copy is made of an assembly, this event occurs for all documents in the assembly, even those that are not copied but stay referenced. This can occur if the **Default Duplicator Action** option of the document type is set to **Reference the source** instead of **Reference the copies** (which were made by the command). You can confirm which documents were actually copied by checking the value of the **TargetFolder** parameter, which will be empty for the documents that were not copied.

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

*InitializeCopy Event

Occurs when documents are copied. Applies to **Copy** and **Paste** commands only.

The **PreInitializeCopy** event can be used to add a confirmation page before the **InitializeCopy** event. **PreInitializeCopy** can also be used as part of a wizard.

Syntax

```
DocCopyMoveEvent_*InitializeCopy(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

Remarks

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

InitializeDerive Event

Occurs when documents are derived. Applies to **Derive** command only.

Syntax

```
DocCopyMoveEvent_InitializeDerive(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

InitializeMove Event

Occurs when documents are moved.

Syntax

```
DocCopyMoveEvent_InitializeMove(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

InitializeReplace Event

Occurs when documents are replaced. Applies to **Replace** command only.

Syntax

`DocCopyMoveEvent_InitializeReplace(`*`Batch`*`, `*`TargetFolder`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

PrepareCopy Event

Occurs before a batch of documents are copied by various Meridian commands, but unlike other initialization events, it occurs for the current document in the batch.

Syntax

`DocCopyMoveEvent_PrepareCopy (`*`Batch`*`, `*`TargetFolder`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder. This event can modify the destination folder.

Example

The following example demonstrates how to assign a project copy to a known project folder (retrieved from a document property, for example) and how to suppress the **Select Folder** and **Reference Explorer** (for reference selection if references exist) dialog boxes.

```
Sub DocCopyMoveEvent_PrepareCopy(Batch, TargetFolder)
  If Not Document Is Nothing Then
    If Client.ImportType = AS_IT_COPIED And Client.ImportDetails =
```

```
AS_ID_CREATEPROJCOPY Then
      'Add your code for document objects
      TargetFolder = "\Design\Projects\894" 'Or some property value
      'Suppress folder selection dialog
      Client.Confirmation(AS_CONFIRM_PROJECT_FOLDER) = False
      'Suppress reference selection dialog
      Client.Confirmation(AS_CONFIRM_SELECT_ASSEMBLY_ITEMS) = False
    End If
  ElseIf Not Folder Is Nothing Then
    'Add your code for folder objects
  End If
End Sub
```

SelectTarget Event

Occurs when a document is copied by the **Import Documents**, **Copy** (simple), **Derive**, and **Replace** commands. This event can be used to show the user a dialog box for folder, projects, or no dialog box.

Syntax

```
DocCopyMoveEvent_SelectTarget (Batch, TargetFolder, DialogToShow)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected. This event occurs once for each document in the batch.
TargetFolder	An object that represents the default destination folder. This event can set the destination folder.
DialogToShow	The type of dialog box to show for folder selection, one of the **AS_SELECT_DIALOG_TYPE** constants.

Remarks

The **Document** object in this event is the source document. This event is equivalent to the **AS_CONFIRM_NO_SELECTPROJECTWIZARD** constant. For the corresponding event for newly created documents, see DocGenericEvent_SelectTarget event.

TerminateCopy Event

Occurs when a copy event is terminated. Applies only to **Copy** and **Paste** commands.

Syntax

`DocCopyMoveEvent_TerminateCopy(`*`Batch`*`, `*`TargetFolder`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

TerminateDerive Event

Occurs when document derive event is terminated. Applies to **Derive** command only.

Syntax

`DocCopyMoveEvent_TerminateDerive(`*`Batch`*`, `*`TargetFolder`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

TerminateMove Event

Occurs when document move event is terminated.

Syntax

```
DocCopyMoveEvent_TerminateMove(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

TerminateReplace Event

Occurs when document replace event is terminated. Applies to **Replace** command only.

Syntax

```
DocCopyMoveEvent_TerminateReplace(Batch, TargetFolder)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.

Document Generic Events

These are events that do not fall under any of the other document event categories.

Document Generic Event Sequences

The events that occur for the document generic commands are shown in the following lists in the sequence that they occur.

Change Document Type event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_PreInitializeChangeDocumentType**
3. **DocGenericEvent_InitializeChangeDocumentType**
4. **DocGenericEvent_PreBeforeChangeDocumentType**
5. **DocGenericEvent_BeforeChangeDocumentType**
6. **DocGenericEvent_AfterChangeDocumentType**
7. **DocGenericEvent_BeforeNewDocument**
8. **DocGenericEvent_OnProperties**
9. **DocGenericEvent_AfterNewDocument**
10. **DocGenericEvent_TerminateChangeDocumentType**

Delete event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializeDelete**
3. **DocGenericEvent_BeforeDelete**
4. **DocGenericEvent_AfterDelete**
5. **DocGenericEvent_TerminateDelete**

Draft Print event sequence

Does not occur in PowerWeb.

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializePrint**
3. **DocGenericEvent_BeforePrint**
4. **DocGenericEvent_AfterPrint**
5. **DocGenericEvent_TerminatePrint**

Issue New Filename event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializeCalculateFileName**
3. **DocGenericEvent_TerminateCalculateFileName**

New Document event sequence

Does not occur when new documents are created by the **Database Import Wizard** in Configurator.

1. **DocGenericEvent_BeforeSelectDocumentType**
2. The following two events DO NOT occur when new documents are created by the **Document Import** tool in PowerUser:
    a. **DocGenericEvent_InitializeNewDocument**
    b. **DocGenericEvent_InitializeCalculateFileName**
3. **DocGenericEvent_SelectTarget**
4. **DocGenericEvent_BeforeNewDocument**
5. The following four events occur ONLY if `Client.Confirmation (AS_CONFIRM_NO_SELECTPROJECTWIZARD) = False` and **Document.ParentFolder** is not a valid folder:
    a. **ProjectWorkflowEvent_PrepareBrowser**
    b. **ProjectWorkflowEvent_InitializeExpandItem**
    c. **ProjectWorkflowEvent_BeforeExpandItem**
    d. **DocGenericEvent_DocumentFolderSelected**
6. **DocGenericEvent_OnProperties**: Does not occur when new documents are created by the Document Import tool in PowerUser.
7. The following four events occur ONLY if `Client.Confirmation (AS_CONFIRM_NO_SELECTPROJECTWIZARD) = True` and **Document.ParentFolder** is not a valid folder.

a. **ProjectWorkflowEvent_PrepareBrowser**
b. **ProjectWorkflowEvent_InitializeExpandItem**
c. **ProjectWorkflowEvent_BeforeExpandItem**
d. **DocGenericEvent_DocumentFolderSelected**

8. **DocGenericEvent_AfterNewDocument**
9. The following two events occur ONLY if an expression is configured for the **Calculate file name** option of the document type:
    a. **DocGenericEvent_BeforeCalculateFileName**
    b. **DocGenericEvent_AfterCalculateFileName**
10. The following two events DO NOT occur when new documents are created by the Document Import tool in PowerUser:
    a. **DocGenericEvent_TerminateCalculateFileName**
    b. **DocGenericEvent_TerminateNewDocument**

Open in Application event sequence

Occurs in PowerWeb when documents are downloaded.

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializeOpenInApplication**
3. **DocGenericEvent_BeforeOpenInApplication**
4. **DocGenericEvent_AfterOpenInApplication**
5. **DocGenericEvent_TerminateOpenInApplication**

Prepare for Offline Work event sequence

Does not occur in PowerWeb.

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializePrepareForOffline**
3. **DocGenericEvent_BeforePrepareForOffline**
4. **DocGenericEvent_AfterPrepareForOffline**
5. **DocGenericEvent_TerminatePrepareForOffline**

Redline event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_BeforeEditRedlines**: Does not occur in PowerWeb.
3. **DocGenericEvent_AfterEditRedlines**

Refresh Thumbnail event sequence

Does not occur in PowerWeb.

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializeUpdateThumbnail**: Does not occur if the **Refresh thumbnail** button on the **Document** page is clicked.
3. **DocGenericEvent_BeforeUpdateThumbnail**
4. **DocGenericEvent_AfterUpdateThumbnail**
5. **DocGenericEvent_TerminateUpdateThumbnail**: Does not occur if the **Refresh thumbnail** button on the **Document** page is clicked.

Rename event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializeRename**
3. **DocGenericEvent_BeforeRename**
4. **DocGenericEvent_AfterRename**
5. **DocGenericEvent_TerminateRename**

Replace Content event sequence

Does not occur in PowerWeb.

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_BeforeReplaceContent**
3. **DocGenericEvent_AfterReplaceContent**

AfterCreateReference Event

Occurs after a reference between documents has been manually created by a user.

Syntax

```
DocGenericEvent_AfterCreateReference(Batch, RefType, TargetDocument,
[RefDisplayName])
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
RefType	Name of the reference type assigned to the reference.
TargetDocument	Document object to which the reference was made.
RefDisplayName	Name that was assigned to the reference. The value may be set as described in DocGenericEvent_BeforeCreateReference event.

AfterDeleteReference Event

Occurs after a reference between documents has been manually deleted by a user.

Syntax

```
DocGenericEvent_AfterDeleteReference(Batch, RefType, TargetDocument,
[RefDisplayName])
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
RefType	Name of the reference type assigned to the reference.
TargetDocument	Document object to which the reference was made.
RefDisplayName	Name that was assigned to the reference.

Remarks

If the **Batch.Abort** method is called in the **DocGenericEvent_BeforeDeleteReference** event from PowerWeb, the reference will not be deleted.

AfterEditRedlines Event

Occurs after documents are redlined by a user with the Meridian viewer.

Syntax

```
DocGenericEvent_AfterEditRedlines(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

AfterReplaceContent Event

Occurs after a user replaces the content of documents.

Syntax

```
DocGenericEvent_AfterReplaceContent(Batch, SourceFile)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
SourceFile	A string containing the path and filename of the source file selected by the user to replace the document's content.

BeforeCreateReference Event

Occurs before a reference between documents is manually created by a user.

Syntax

```
DocGenericEvent_BeforeCreateReference(Batch, RefType, TargetDocument,
[RefDisplayName])
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
RefType	Name of the reference type that will be assigned to the reference.
TargetDocument	Document object to which the reference will be made.
RefDisplayName	Default name that will be assigned to the reference. This parameter is not supplied in all cases. The value may be set in this event procedure.

BeforeDeleteReference Event

Occurs before a reference between documents is manually deleted by a user.

Syntax

```
DocGenericEvent_BeforeDeleteReference(Batch, RefType, TargetDocument,
[RefDisplayName])
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
RefType	Name of the reference type assigned to the reference.
TargetDocument	Document object to which the reference was made.
RefDisplayName	Name that was assigned to the reference.

Remarks

If the **Batch.Abort** method is called in this event from PowerWeb, the reference will not be deleted.

BeforeEditRedlines Event

Occurs before documents are redlined by a user with the Meridian viewer.

Syntax

```
DocGenericEvent_BeforeEditRedlines(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

BeforeReplaceContent Event

Occurs before a user replaces the content of documents.

Syntax

```
DocGenericEvent_BeforeReplaceContent(Batch, SourceFile)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
SourceFile	A string containing the path and filename of the source file selected by the user to replace the document's content.

Remarks

Importing a document using drag and drop on a released document will create a new revision. Depending on your organization's business process for this action, the document might need to be placed in different workflow state, for example, for review before the new content can be released. We recommend that you do that in this event.

If this event handler is not customized, an **Access is denied** error will result and the action will fail. If the document should remain in the released state after the import, this event handler should still be implemented to place the document in a different workflow state and then the document released in the **DocGenericEvent_AfterNewDocument** event handler.

By default, the redlines for the existing document content will be deleted when the content is replaced. If the redlines should be retained (to check that changes to the document have been done properly, for example), you can prevent the redlines from being deleted by setting the **AS_CONFIRM_CLEANREDLINESONREPLACECONTENT** constant to **False** in this event. The default is **True**.

BeforeSelectDocType Event

Outputs the list of available document types to show the user based on the type of operation being carried out such as a new document or change the document type.

Syntax

```
DocGenericEvent_BeforeSelectDocumentType(Batch, DocTypesList)
```

Parameters

Name	Description
Batch	An object that represents the action the user has selected.
DocTypesList	A semicolon-delimited list of available document types for the action being carried out.

Remarks

This functionality is only available in PowerWeb. On wizard pages, this event is fired once.

*CalculateFilename Events

Occurs when the filename of a document is calculated by its document type.

Syntax

```
DocGenericEvent_*CalculateFilename(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

*ChangeDocumentType Events

Occurs when a user changes the document type of documents. The **PreInitializeChangeDocumentType** and **PreBeforeChangeDocumentType** events can be used to add confirmation pages before the **InitializeChangeDocumentType** and

BeforeChangeDocumentType events. These events can also be used as part of a wizard.

Syntax

```
DocGenericEvent_*ChangeDocumentType(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
NewType	Passed only to the **PreBeforeChangeDocumentType** and **BeforeChangeDocumentType** procedures because at that moment the document type has not yet been changed.
OldType	Passed only to the **AfterChangeDocumentType** procedure because the document type has already been changed.

Remarks

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

*Delete Events

Occurs when documents or folders are deleted by a user.

Syntax

```
DocGenericEvent_*Delete(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of objects the user has selected.

Remarks

When a user deletes a folder, the **Document** object is **Nothing**. When a user deletes a document, the **Document** object is the document being deleted. When a user deletes a folder that contains one or more documents, the events occur once for each document and then once for the parent folder.

Example

```
Sub DocGenericEvent_BeforeDelete(Batch)
  If Not Document Is Nothing Then
    Batch.PrintDetails "Before delete " + Document.FileName
  ElseIf Not Folder Is Nothing Then
    Batch.PrintDetails "Before delete " + Folder.Name
  End If
End Sub
```

DocumentFolderSelected Event

Occurs after the project folder selection dialog box has been shown to the user to select the destination folder for a new project copy document. Also shown to select a master document folder when releasing new project documents as new master documents.

The project folder selection dialog box can be controlled with the **ProjectWorkflowEvent_PrepareBrowser**, **ProjectWorkflowEvent_InitializeExpandItem**, and **ProjectWorkflowEvent_BeforeExpandItem** event procedures described in Project Workflow Events.

Syntax

`DocGenericEvent_DocumentFolderSelected (`*`Batch`*`, `*`SelectedFolder`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of master documents the user has selected.
SelectedFolder	The name of the folder selected by the user.

Remarks

If `Client.Confirmation (AS_CONFIRM_NO_SELECTPROJECTWIZARD)` is **False** after the **DocGenericEvent_BeforeNewDocument** event and **Document.ParentFolder** is not a valid folder at the time that the new document properties must be set, the folder selection dialog box is shown before the **DocGenericEvent_OnProperties** event in order to set the destination folder.

If `Client.Confirmation (AS_CONFIRM_NO_SELECTPROJECTWIZARD)` is **True** after the **DocGenericEvent_OnProperties** event and **Document.ParentFolder** is still not a valid folder, the folder selection dialog box is shown before the **DocGenericEvent_AfterNewDocument** event.

The selected folder is set as the **Folder** object in the **DocGenericEvent_OnProperties** and **DocGenericEvent_AfterNewDocument** events.

*NewDocument Events

Occurs when documents are created by a user with any method (template, import, Application Integration). Also when a document is moved to a different folder or changed to a different document type.

Syntax

```
DocGenericEvent_*NewDocument(Batch, Action, SourceFile, DocType,
DocTemplate)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Action	A long integer that represents one or more **AS_IMPORTTYPE** constants.
SourceFile	A string that contains the name of the source file imported as the new document.
DocType	An object that represents the document type selected by the user for the new document.
DocTemplate	A string that contains the name of the document template selected by the user.

Remarks

We recommend not setting any property values in the **BeforeNewDocument** event as this might cause the document to be created in a temporary parent folder before it is moved to its final destination, which might cause conflicts with existing documents.

The **Before** and **After** procedures are invoked when documents are imported with the Document Import tool. The **Initialize** and **Terminate** procedures are not invoked by the Document Import tool and are invoked only once per batch by the PowerWeb**Import Documents** command. The **FailCurrent** and **Abort** methods of the **Batch** object are also available; however, the **BatchIndex** property will always be 1.

When documents are imported using drag-and-drop, the source of the files cannot be determined by Meridian so the value of the **Client.ClientID** property is **AS_CID_POWERUSER** regardless of the source of the documents. For your script to know where the documents originated from to perform validation or other processing, the users must use the file commands that are provided by the Meridian application links.

Example

You can test the value of the **$$ICIMP_PropInvalid** batch argument to determine if property validation failed while running the Document Import tool. If the validation failed, then you can prevent the import of the failed document using code similar to the following example.

```
Sub DocGenericEvent_BeforeNewDocument(Batch, Action, SourceFile,
DocType, DocTemplate)
   If CBool(Batch.Argument("$$ICIMP_PropInvalid")) = True Then
       Batch.FailCurrent("Property validation failed")
   End If
End Sub
```

Use the object argument `Batch.Argument("__$$RelatedProjectCopy")` to obtain the project copy that is being released as a master document. It can be useful to calculate the location for the master document location if it is being created for the first time.

Use the object argument `Vault.Argument("__$$RelatedTransmittal")` to relate a transmittal to a new submittal. Use the object argument `Vault.Argument("__$$SubmittalSender")` to relate a person to a new submittal.

For more information, see Object Arguments.

OnProperties Event

Occurs when a user views or edits the property pages of a document or folder.

Syntax

```
DocGenericEvent_OnProperties(Command, Abort)
```

Parameters

Name	Description
Command	A long integer that represents one or more **AS_PROP_CMD** constants that indicate which button the user clicked. The constants are described in the following table.
Abort	Set to **True** to abort the operation.

Remarks

This event occurs in the Meridian client applications when **Command** includes the constants indicated in the following table.

This AS_PS_CMD_APPLY constant is represented in PowerUser only when the **Finish** button is clicked at the end of a wizard. In PowerWeb, it is included every time the **Next** button is clicked for a wizard page. Property values may not be set in PowerWeb if this constant is represented.

This event does not occur in PowerWeb when creating folders.

AS_PROP_CMD constants

Constant	Action	PowerWeb	PowerUser
AS_PS_CMD_VIEW	The user started viewing the property page.	**Not Included**	Included
AS_PS_CMD_EDIT	The user clicked **Edit** to start property page (wizard) editing.	Included	Included
AS_PS_CMD_APPLY	The user clicked **Next** to apply changes to the current page.	Included	Included
AS_PS_CMD_FINISH	The user clicked **Finish** to apply changes to the last page and stop property page (wizard) editing.	Included	Included
AS_PS_CMD_CANCEL	The user clicked **Cancel** to discard changes and stop property page (wizard) editing.	**Not Included**	Included

Use this event with much caution. In PowerUser, when **Command** includes **AS_PS_CMD_VIEW**, setting **Abort** to **True** will prevent display of all property pages. Attempting to set property values at that time can cause errors such as "**The object is currently being edited by another session**" and lost data.

We recommend that you not execute other actions, make modifications to the document content, invoke anything that takes a relatively long time, or invoke any user interface functions during this event. Such actions can have unexpected or unpredictable results.

*OpenInApplication Events

Occurs when documents are opened by a user in their native application.

Syntax

```
DocGenericEvent_*OpenInApplication(Batch, AppName)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
AppName	A string that contains the name of the application in which the documents will be opened. This is the name of the application registered in Windows on the user's computer for the file type of the documents.

PrepareCommand Event

Occurs after a user selects a document command on the shortcut menu to act upon the selected document.

Syntax

```
DocGenericEvent_PrepareCommand(Batch, CmdID)
```

Parameters

Name	Description
Batch	An object that represents the current batch of documents.
CmdID	A string that identifies the internal name of the command. For the identifiers that may be used, see Meridian Enterprise command identifiers.

Remarks

Not available in PowerWeb. This event is intended to be used to set the **Batch.ProcessAsBatch** property to **True** before programmatically adding documents with the **Batch.AddDocuments** method to a batch in a custom command's **Initialize** event.

*PrepareForOffline Events

Occurs when a user prepares documents for offline mode operation.

Syntax

```
DocGenericEvent_*PrepareForOffline(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

*Print Events

Occurs when documents are printed by a user from Meridian with the viewer. For information on configuring watermark printing with the **Watermark** properties, see *Configure Watermark Printing* in the *Meridian Enterprise Configuration Guide*.

Syntax

```
DocGenericEvent_*Print(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Remarks

The `DocGenericEvent_AfterPrint` event does not occur in PowerWeb.

*Rename Events

Occurs when documents or folders are renamed by a user.

Syntax

```
DocGenericEvent_*Rename(Batch, NewName)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
NewName	A string that contains the new name entered by a user.

SelectTarget Event

Occurs when a new document is created by the **Create New** and **Add Document** commands. This event can be used to show the user a dialog box for folder, projects, or no dialog box.

Syntax

```
DocGenericEvent_SelectTarget (Batch, TargetFolder, DialogToShow)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected. This event occurs once for each document in the batch.
TargetFolder	An object that represents the destination folder. This event can set the destination folder.
DialogToShow	The type of dialog box to show for project selection, one of the **AS_SELECT_DIALOG_TYPE** constants.

Remarks

The **Document** object in this event is the new document. This event is equivalent to the **AS_CONFIRM_NO_SELECTPROJECTWIZARD** constant. For the corresponding event for copied documents, see DocCopyMoveEvent_SelectTarget event.

*UpdateThumbnail Events

Occurs when a user updates the thumbnail images of documents manually.

Syntax

```
DocGenericEvent_*UpdateThumbnail(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

DraftPrint_SetWatermark Event

Occurs when documents are printed by a user from Meridian Explorer or PowerWeb with the viewer. For information on configuring watermark printing with the **Watermark** properties, see the *Configure Watermark Printing* article in the *Meridian Enterprise Server Administrator's Guide*.

Support for PowerWeb was added in the 2022 release.

Syntax

```
DraftPrint_SetWatermark
```

Remarks

The settings **Enable draft print** and **Enable watermarks on draft print hardcopies** must be set to **True** as described in the *Configure the Application Options* article in the *Meridian Enterprise Server Administrator's Guide*.

Document Hybrid Events

Document hybrid events allow you to extend custom functionality when hybrid parts are attached and detached.

Hybrid parts event sequence

The events that occur for the document hybrid part commands are shown in the following list in the sequence that they occur.

- **DocGenericEvent_PrepareCommand**
- **DocHybridEvent_InitializeHybridParts**
- **DocHybridEvent_BeforeHybridPart**
- **DocHybridEvent_AfterHybridPart**
- **DocHybridEvent_TerminateHybridParts**
- **DocGenericEvent_*Rename events**: Occurs in PowerWeb only when creating a new part.

AfterHybridPart Event

Occurs after a hybrid document action is performed on a document.

Syntax

```
DocHybridEvent_AfterHybridPart(Batch, Action, PartName)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Action	A long integer that represents one or more **AS_HYBRID_ACTION** constants.
PartName	Optional name for the hybrid part.

BeforeHybridPart Event

Occurs before a hybrid document action is performed on a document.

Syntax

```
DocHybridEvent_BeforeHybridPart(Batch, Action, PartName)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Action	A long integer that represents one or more **AS_HYBRID_ACTION** constants.
PartName	Optional name for the hybrid part.

InitializeHybridParts Event

Occurs before hybrid document actions are performed on a batch of documents.

Syntax

```
DocHybridEvent_InitializeHybridParts(Batch, Action)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Action	A long integer that represents one or more **AS_HYBRID_ACTION** constants.

BeforeConvert2Hybrid Event

Occurs before a normal document is converted into a hybrid document by the **Convert to Hybrid** command.

Syntax

```
DocHybridEvent_BeforeConvert2Hybrid(Batch, TemplateName)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TemplateName	The name of the template selected by the user.

AfterConvert2Hybrid Event

Occurs after a normal document has been converted into a hybrid document by the **Convert to Hybrid** command.

Syntax

```
DocHybridEvent_AfterConvert2Hybrid(Batch, TemplateName)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TemplateName	The name of the template selected by the user.

Document Project Copy Events

Document project copy events occur when documents created as project copies are modified of using the various Meridian Enterprise commands. Like other document events, these events also receive a **Batch** object. For more information about the **Batch** object, see Batch Object.

Document Project Copy Event Sequences

The events that occur for the document project copy commands are shown in the following lists in the sequence that they occur.

Create Project Copy event sequence

1. **DocCopyMoveEvent_PrepareCopy**
2. **ProjectWorkflowEvent_PrepareBrowser**
3. **ProjectWorkflowEvent_InitializeExpandItem**
4. **ProjectWorkflowEvent_BeforeExpandItem**
5. **DocCopyMoveEvent_InitializeCopy**
6. **DocGenericEvent_InitializeNewDocument**
7. **DocGenericEvent_InitializeCalculateFileName**
8. **DocCopyMoveEvent_BeforeCopyWithReferences**
9. **DocProjectCopyEvent_ProjectCopyExist**
10. **DocCopyMoveEvent_BeforeCopy**
11. **DocCopyMoveEvent_AfterCopy**
12. **DocGenericEvent_BeforeNewDocument**
13. **DocGenericEvent_OnProperties**
14. **DocGenericEvent_AfterNewDocument**
15. **DocCopyMoveEvent_AfterCopyWithReferences**
16. **DocGenericEvent_TerminateCalculateFileName**
17. **DocGenericEvent_TerminateNewDocument**
18. **DocCopyMoveEvent_TerminateCopy**

Confirm Merged with Master event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocProjectCopyEvent_InitializeConfirmMerged**
3. **DocProjectCopyEvent_BeforeConfirmMerged**
4. **DocProjectCopyEvent_AfterConfirmMerged**
5. **DocProjectCopyEvent_TerminateConfirmMerged**

Confirm Superseded by Master event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocProjectCopyEvent_InitializeConfirmSuperseded**
3. **DocProjectCopyEvent_BeforeConfirmSuperseded**
4. **DocProjectCopyEvent_AfterConfirmSuperseded**
5. **DocProjectCopyEvent_TerminateConfirmSuperseded**

Discard from Project event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocProjectCopyEvent_InitializeDiscardFromProject**
3. **DocProjectCopyEvent_BeforeDiscardFromProject**
4. **DocProjectCopyEvent_AfterDiscardFromProject**
5. **DocProjectCopyEvent_TerminateDiscardFromProject**

Link to Master event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocProjectCopyEvent_BeforeLinkToMaster**
3. **DocProjectCopyEvent_AfterLinkToMaster**

Release as Master Revision event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocProjectCopyEvent_PreInitializeReleaseToMaster**
3. **DocProjectCopyEvent_InitializeReleaseToMaster**
4. The following two events ONLY occur if the master document does not yet exist:
    a. **DocGenericEvent_BeforeNewDocument**
    b. **DocGenericEvent_OnProperties**
5. The following four events ONLY occur if the master document does not yet exist AND if a parent folder has not yet been specified:
    a. **ProjectWorkflowEvent_PrepareBrowser**
    b. **ProjectWorkflowEvent_InitializeExpandItem**
    c. **ProjectWorkflowEvent_BeforeExpandItem**
    d. **DocGenericEvent_DocumentFolderSelected**
6. **DocGenericEvent_AfterNewDocument**: Occurs only if the master document does not yet exist.
7. The following two events ONLY occur if the master document does not yet exist AND an expression is configured for the **Calculate file name** option of the document type:
    a. **DocGenericEvent_BeforeCalculateFileName**
    b. **DocGenericEvent_AfterCalculateFileName**
8. **DocProjectCopyEvent_PreBeforeReleaseToMaster**
9. **DocProjectCopyEvent_BeforeReleaseToMaster**
10. The following two events ONLY occur if the project copy has been modified:
    a. **DocProjectCopyEvent_BeforeMasterUpdate**
    b. **DocProjectCopyEvent_AfterMasterUpdate**
11. **DocProjectCopyEvent_AfterReleaseToMaster**
12. **DocProjectCopyEvent_TerminateReleaseToMaster**

AfterLinkToMaster Event

Occurs when the user runs the **Link to Master Document** command.

Syntax

```
DocProjectCopyEvent_AfterLinkToMaster (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied.

Remarks

The **Document** object is the project copy during this event.

AfterMasterUpdate Event

Occurs after a new revision of a master document is updated from a project copy.

Syntax

```
DocProjectCopyEvent_AfterMasterUpdate (Batch, ProjectCopyDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
ProjectCopyDoc	An object that represents the project copy document.

AfterTransferToNext Event

Occurs when the user runs the **Transfer to Next** command.

Syntax

```
DocProjectCopyEvent_AfterTransferToNext (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied.

Remarks

The **Document** object is the project copy during this event.

AfterUnlinkFromMaster Event

Occurs when the user runs the **Unlink from Master Document** command.

Syntax

```
DocProjectCopyEvent_AfterUnlinkFromMaster (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied.

Remarks

The **Document** object is the project copy during this event.

BeforeLinkToMaster Event

Occurs when the user runs the **Link to Master Document** command.

Syntax

```
DocProjectCopyEvent_BeforeLinkToMaster (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied.

Remarks

The **Document** object is the project copy during this event.

BeforeMasterUpdate Event

Occurs before a new revision of a master document is updated from a project copy.

Syntax

```
DocProjectCopyEvent_BeforeMasterUpdate (Batch, ProjectCopyDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
ProjectCopyDoc	An object that represents the project copy document.

BeforeTransferToNext Event

Occurs when the user runs the **Transfer to Next** command.

Syntax

```
DocProjectCopyEvent_BeforeTransferToNext (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied.

Remarks

The **Document** object is the project copy during this event.

BeforeUnlinkFromMaster Event

Occurs when the user runs the **Unlink from Master Document** command.

Syntax

```
DocProjectCopyEvent_BeforeUnlinkFromMaster (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied.

Remarks

The **Document** object is the project copy during this event.

*ConfirmMerged Events

Occur when the user runs the **Confirm Merged with Master** command.

Syntax

```
DocProjectCopyEvent_*ConfirmMerged (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied. This parameter is passed only to the **Before** and **After** events.

Remarks

The **Document** object is the project copy during this event.

*ConfirmSuperseded Events

Occur when the user runs the **Confirm Superseded by Master** command.

Syntax

```
DocProjectCopyEvent_*ConfirmSuperseded (Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied. This parameter is passed only to the **Before** and **After** events.

Remarks

The **Document** object is the project copy during this event.

*DiscardFromProject Events

Occur when a project copy is discarded from a project workflow.

Syntax

```
DocProjectCopyEvent_*DiscardFromProject (Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

PreInitializeReleaseToMaster

This event is used to add a confirmation page before the InitializeReleaseToMaster event. This event can be used as part of a wizard.

Syntax

```
DocCopyMoveEvent_PreInitializeReleaseToMaster(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Remarks

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

PreBeforeReleaseToMaster

This event is used to add a confirmation page before the BeforeReleaseToMaster event. This event can be used as part of a wizard.

Syntax

```
DocCopyMoveEvent_PreBeforeReleaseToMaster(Batch, MasterDoc,
ProjectCopyChanged)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied. This parameter is passed only to the **Before** and **After** events.
ProjectCopyChanged	True if the project copy has been changed since it was copied from the master document. This parameter is passed only to the **Before** and **After** events.

Remarks

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

ProjectCopyExist Event

Occurs when an attempt is made to create a project copy and another project copy already exists. The new project copy is not made and an error message is shown to the user.

Syntax

```
DocProjectCopyEvent_ProjectCopyExist(Batch, ProjectCopy)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
ProjectCopy	An object that represents the existing project copy document.

Remarks

The **Document** object is the source document during this event. Only changes made during this event are saved. Changes made in prior events are discarded.

*ReleaseToMaster Events

Occur when a project copy is released as a new revision of the master document from which it was copied.

Syntax

```
DocProjectCopyEvent_*ReleaseToMaster (Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied. This parameter is passed only to the **Before** and **After** events.
ProjectCopyChanged	True if the project copy has been changed since it was copied from the master document. This parameter is passed only to the **Before** and **After** events.

*RequireMerge Events

Occur when the user runs the **Require Merge** command.

Syntax

```
DocProjectCopyEvent_*RequireMerge(Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied. This parameter is passed only to the **Before** and **After** events.

Remarks

The **Document** object is the project copy during this event.

*UndoMakeObsolete Events

Occur when the user runs the **Undo Make Obsolete** command.

Syntax

```
DocProjectCopyEvent_*UndoMakeObsolete (Batch, MasterDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
MasterDoc	An object that represents the master document from which the selected document was copied. This parameter is passed only to the **Before** and **After** events.

Remarks

The **Document** object is the project copy during this event.

ValidateTargetFolder Event

Occurs when an attempt is made to create a project copy. The target folder can be evaluated by script and the user prevented from proceeding. If the folder is not valid, the *ErrorMessage* is displayed.

Syntax

```
DocProjectCopyEvent_ProjectCopyExist(Batch, TargetFolder,
ErrorMessage)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
TargetFolder	An object that represents the destination folder.
ErrorMessage	An object that represents the error message to be displayed.

Remarks

This event is not supported when **ActiveX compatibility mode** is enabled. For more information, see the *Personal Preferences* article in the *Meridian Enterprise User's Guide*.

Examples

Following are example implementations of this event.

```
Sub DocProjectCopyEvent_ValidateTargetFolder(Batch, TargetFolder, ErrorMessage)
  If InStr (TargetFolder.Name, "Project") = 0 Then
    ErrorMessage = "You can create project copies only in projects."
  End If
End Sub
```

Example that gets the project name.

```
Sub DocProjectCopyEvent_ValidateTargetFolder(Batch, TargetFolder, ErrorMessage)
    Dim sProjectName
    If TargetFolder.IsProject Then sProjectName = TargetFolder.Name Else sProjectName = TargetFolder.ParentProject.Name
    If InStr (TargetFolder.Name, "Master") > 0 And Not TargetFolder.ParentProject Is Nothing Then
        ErrorMessage = ""
    Else
        ErrorMessage = "Please select the 'Masters' folder in " & sProjectName & "."
    End If
End Sub
```

If you want to check if the parent project is in a project workflow:

```
Sub DocProjectCopyEvent_ValidateTargetFolder(Batch, TargetFolder, ErrorMessage)
  If TargetFolder.ParentProject.WorkFlowState = AS_PWF_RELEASED Then
      If TargetFolder.ParentProject.WorkFlowState = AS_PWF_RELEASED Then
          ErrorMessage = "Please select another project; the current project status is 'Closed'."
      End If
  End If
End Sub
```

Document Type Workflow Events

The events in the **DocWorkflowEvent** category allow you to customize the actions performed by the workflows configured for document types.

Document Type Workflow Event Sequences

The events that occur for the document type workflow commands are shown in the following lists in the sequence that they occur.

Change To-Do Person event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocWorkflowEvent_InitializeChangeWFPerson**
3. **DocWorkflowEvent_BeforeChangeWFPerson**
4. **DocWorkflowEvent_AfterChangeWFPerson**
5. **DocWorkflowEvent_TerminateChangeWFPerson**

Change Workflow Manager event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocWorkflowEvent_InitializeChangeWFManager**
3. **DocWorkflowEvent_BeforeChangeWFManager**
4. **DocWorkflowEvent_AfterChangeWFManager**
5. **DocWorkflowEvent_TerminateChangeWFManager**

Execute transition event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocWorkflowEvent_InitializeChangeWFState**
3. **DocWorkflowEvent_BeforeChangeWFState**
4. **DocWorkflowEvent_AfterChangeWFState**
5. **DocWorkflowEvent_TerminateChangeWFState**

Revoke event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocWorkflowEvent_PreInitializeRevokeWF**
3. **DocWorkflowEvent_InitializeRevokeWF**
4. **DocWorkflowEvent_PreBeforeRevokeWF**
5. **DocWorkflowEvent_BeforeRevokeWF**
6. **DocWorkflowEvent_AfterRevokeWF**
7. **DocWorkflowEvent_TerminateRevokeWF**

*ChangeWFManager Events

Occurs when the manager of a document type workflow is manually changed to a different user.

Syntax

```
DocWorkflowEvent_*ChangeWFManager(Batch, Person)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Person	An object that represents the user that is selected as the new workflow manager.

*ChangeWFPerson Events

Occurs when the to-do person of a document type workflow is manually changed to a different user.

Syntax

```
DocWorkflowEvent_*ChangeWFPerson(Batch, Person, Comment)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Person	An object that represents the user that is selected as the new workflow manager.
Comment	A string that contains the comment entered by the user that is assigning the to-do person.

*ChangeWFState Events

Occurs when a document type workflow is routed to a different state.

Syntax

```
DocWorkflowEvent_*ChangeWFState(Batch, TargetState, Person, Comment)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
SourceState	A long integer that represents one or more **AS_WF_STATE** constants. Passed only to the **Before** and **After** events.
TargetState	A long integer that represents one or more **AS_WF_STATE** constants.
Person	An object that represents the user to whom the documents are being assigned.
Comment	A string containing the comment entered by the user.

*RevokeWF Events

Occurs when a document type workflow is revoked by a user.

Syntax

```
DocWorkflowEvent_*RevokeWF (Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Remarks

The **PreBeforeRevokeWF** event is used to add a confirmation page before the revoke a document workflow action, and the **PreInitializeRevokeWF** event is used to add a confirmation page before initializing the action.

Learn how to create confirmation pages. You can also see examples of this functionality in this Knowledgebase article.

AfterReplaceReleased Event

Occurs for a document after a document that replaced it has been released from a workflow.The released document must have been made by the **Replace Document** command.

Syntax

```
AfterReplaceReleased (Batch, ReplacingDoc)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
ReplacingDoc	An object that represents the document that is being released.

Remarks

This event occurs after all other events for the document. It can be used to perform such actions as:

- Sending email notification of the release of the replacement
- Flagging the replaced document for archiving
- Removing the replaced document from the Data Library repository

- Setting a custom status property value on the replaced document
- Moving the replaced document to another location

Document Working Copy Events

The document working copy events occur only if the Work Isolation Mode of the vault is enabled, and they occur instead of the document type workflow events.

Work Isolation Mode is a legacy feature. We still support customers who have it enabled, but we do not allow it to be enabled in new Vaults.

Document Working Copy Event Sequences

The events that occur for the document working copy commands are shown in the following lists in the sequence that they occur.

Create Working Copy event sequence

- DocGenericEvent_PrepareCommand
- DocWorkingCopyEvent_BeforeCreateWC
- DocWorkingCopyEvent_AfterCreateWC
- DocWorkingCopyEvent_TerminateCreateWC
- **DocGenericEvent_*OpenInApplication** events: In PowerWeb only if the Accruent Upload/Download Control option **Open in application** is enabled

Revoke Working Copy event sequence

- DocGenericEvent_PrepareCommand
- DocWorkingCopyEvent_InitializeRevokeWC
- DocWorkingCopyEvent_BeforeRevokeWC
- DocWorkingCopyEvent_AfterRevokeWC
- DocWorkingCopyEvent_TerminateRevokeWC

Submit Working Copy event sequence

- DocGenericEvent_PrepareCommand
- DocWorkingCopyEvent_InitializeSubmitWC
- DocWorkingCopyEvent_BeforeSubmitWC

- DocWorkingCopyEvent_AfterSubmitWC
- DocWorkingCopyEvent_TerminateSubmitWC

*CreateWC Events

Occurs when a user creates working copies of documents.

Syntax

```
DocWorkingCopyEvent_*CreateWC(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

*RevokeWC Events

Occurs when a user revokes working copies of documents.

Syntax

```
DocWorkingCopyEvent_*RevokeWC(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

*SubmitWC Events

Occurs when a user submits working copies of documents.

Syntax

```
DocWorkingCopyEvent_*SubmitWC(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Folder Generic Events

The folder generic events allow you to implement custom functionality for folders, including folder types. Some of the events that apply to folders also apply to documents.

Folder Generic Event Sequences

The events that occur for the folder generic commands are shown in the following lists in the sequence that they occur.

Change Folder Type event sequence

1. **DocGenericEvent_PrepareCommand**
2. **FolderGenericEvent_InitializeChangeFolderType**
3. **FolderGenericEvent_BeforeChangeFolderType**
4. **FolderGenericEvent_AfterChangeFolderType**
5. **FolderGenericEvent_BeforeNewFolder**
6. **FolderGenericEvent_AfterNewFolder**
7. **FolderGenericEvent_TerminateChangeFolderType**

Delete Folder event sequence

1. **DocGenericEvent_PrepareCommand**
2. **DocGenericEvent_InitializeDelete**
3. **DocGenericEvent_BeforeDelete**
4. **DocGenericEvent_AfterDelete**
5. **DocGenericEvent_TerminateDelete**

New Folder event sequence

1. **FolderGenericEvent_BeforeNewFolder**
2. **DocGenericEvent_OnProperties**
3. **FolderGenericEvent_AfterNewFolder**
4. **DocGenericEvent_*Rename** events: Occurs in PowerWeb only when creating a new folder.

AfterNewFolder Event

Occurs after a user creates a new folder in the vault.

Syntax

```
FolderGenericEvent_AfterNewFolder(Batch, Action)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Action	Long integer that represents one or more **AS_IMPORTTYPE** constants.

BeforeNewFolder Event

Occurs before a user creates a new folder in the vault.

Syntax

```
FolderGenericEvent_BeforeNewFolder(Batch, Action)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Action	Long integer that represents one or more **AS_IMPORTTYPE** constants.

*ChangeFolder Events

Occurs when a user changes the folder type of an existing folder.

Syntax

```
FolderGenericEvent_*ChangeFolder(Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
NewType	A string that contains the name of the new folder type selected by the user. Passed only to the **BeforeChangeFolder** event.
OldType	A string that contains the name of the original folder type of the folder. Passed only to the **AfterChangeFolder** event.

FileUploaded Event

Occurs when a file is uploaded, but *before* a document is created or updated.

Syntax

```
FolderGenericEvent_FileUploaded (Batch, FileName As String, FilePath
As String, CancelDefault as Boolean)
```

Parameters

Name	Description
Batch	The current batch object.
FileName	The display name of the uploaded file.
FilePath	The local path of the uploaded file on the server PC.
CancelDefault	If set to **True**, the event is canceled. If set to **False**, the event is not canceled.

Package Events

The package events occur during the life cycle of export and import packages in Meridian Enterprise Server.

ExportPackage_ChangeState Event

Occurs when the status of an export package is changed by a user.

Syntax

```
ExportPackage_ChangeState(Package, PreviousState, User)
```

Parameters

Name	Description
Package	An object that represents the export package.
PreviousState	The previous state of the export package represented by one of the **AS_EXPORTPACKAGE_STATUS_VALUES** constants.
User	The name of the user that changed the package status.

Remarks

This event is intended for use with the Vault.SendNotification method to alert appropriate users about documents exported from a folder. Because that method requires a **Document** object context, pass it a document object and not the package object.

Example

```
Sub ExportPackage_ChangeState(Package, PreviousState, User)
  Dim doc
  'GlobalID of a document related to the package
  Set doc = Vault.GetDocument("{0F002100-998A-11E5-0000-
56598CDC4B23}")
  doc.Log "The export package named " & package & " was changed from
the " & _
  PreviousState & " state by " & User
  Vault.SendNotification "PACKAGEMAIL", doc
End Sub
```

ImportPackage_AfterImportedFromPortal Event

Occurs after a document is imported from Meridian Portal.

Syntax

```
ImportPackage_AfterImportedFromPortal ()
```

ImportPackage_AfterReadProperties Event

Occurs after the properties are read for a document that was imported from an import package.

Syntax

```
ImportPackage_AfterReadProperties (Batch, Package)
```

Parameters

Name	Description
Batch	An object that represents the documents contained in the import package.
Package	An object that represents the import package.

ImportPackage_ChangeState Event

Occurs when the status of an import package is changed by a user.

Syntax

```
ImportPackage_ChangeState(Package, PreviousState, User)
```

Parameters

Name	Description
Package	An object that represents the import package.
PreviousState	The previous state of the import package represented by one of the **AS_IMPORTPACKAGE_STATUS_VALUES** constants.
User	The name of the user that changed the package status.

Remarks

This event is intended for use with the Vault.SendNotification method to alert appropriate users about documents imported from a package. Because that method requires a **Document** object context, pass it a document object and not the package object.

Example

```
Sub ImportPackage_ChangeState(Package, PreviousState, User)
  Dim doc
  'GlobalID of a document related to the package
  Set doc = Vault.GetDocument("{0F002100-998A-11E5-0000-
56598CDC4B23}")
  doc.Log "The import package named " & package & " was changed from
the " & _
  PreviousState & " state by " & User
  Vault.SendNotification "PACKAGEMAIL", doc
End Sub
```

Project Workflow Events

Project workflow events occur when a project workflow is active. Like other document events, these events also receive a **Batch** object. For more information about the **Batch** object, see Batch Object.

Project Workflow Event Sequences

The events that occur for the project workflow commands are shown in the following lists in the sequence that they occur.

The Execute Transition event sequence and the Reroute Project event sequence do not occur in PowerWeb.

Execute Transition event sequence

- **DocGenericEvent_PrepareCommand**
- **ProjectWorkflowEvent_InitializeExecuteTransition**
- **ProjectWorkflowEvent_InitializeWizard**
- **DocGenericEvent_OnProperties**
- **ProjectWorkflowEvent_TerminateWizard**
- **ProjectWorkflowEvent_BeforeExecuteTransition**
- **ProjectWorkflowEvent_AfterExecuteTransition**
- **ProjectWorkflowEvent_TerminateExecuteTransition**

Reassign Managers event sequence

- **DocGenericEvent_PrepareCommand**
- **ProjectWorkflowEvent_InitializeChangeWFManager**
- **ProjectWorkflowEvent_BeforeChangeWFManager**
- **ProjectWorkflowEvent_AfterChangeWFManager**
- **ProjectWorkflowEvent_TerminateChangeWFManager**

Reroute Project event sequence

- **DocGenericEvent_PrepareCommand**
- **ProjectWorkflowEvent_InitializeReroute**
- **ProjectWorkflowEvent_BeforeReroute**
- **ProjectWorkflowEvent_AfterReroute**
- **ProjectWorkflowEvent_TerminateReroute**

*ChangeManager Events

Occurs when a project copy is discarded from a project workflow.

Syntax

```
DocProjectCopyEvent_*DiscardFromProject (Batch)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Manager	An object that represents the user who is the manager of the workflow.
Comment	A string containing the text typed by the user who is changing the workflow manager.

*ExecuteTransition Events

Occurs when a project workflow transition is executed.

Syntax

```
ProjectWorkflowEvent_*ExecuteTransition (Batch, Transition, Manager,
Comment)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Name	Description
Transition	A **WorkflowTransition** object that represents the transition to run. For more information about the **WorkflowTransition** object, see WorkflowTransition Object.
Manager	An object that represents the user who is the manager of the workflow.
Comment	A string containing the text typed by the user who is changing the workflow manager.

*Reroute Events

Occurs when a project workflow is rerouted to a selected state.

Syntax

```
ProjectWorkflowEvent_*ExecuteTransition (Batch, Transition, Manager,
Comment)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
State	A **WorkflowState** object that represents the transition to run. For more information about the **WorkflowState** object, see WorkflowState Object.
Manager	An object that represents the user who is the manager of the workflow.
Comment	A string containing the text typed by the user who is changing the workflow manager.

BeforeExpandItem Event

Occurs before the folder selection dialog box is shown to the user to select a destination folder for a project copy. This event does not occur in PowerWeb.

Syntax

```
ProjectWorkflowEvent_InitializeExpandItem (SubItems)
```

Parameters

Name	Description
SubItems	A variant array with an item for each subfolder of the folder to be expanded. Each element is a variant array with the following elements: 1. ID of the subfolder 2. Display name of the subfolder 3. Combination of **AS_PRJITEM_FLAGS** constants for the item. The initial value is **AS_PRJITEM_MODE_VISIBLE** + **AS_PRJITEM_MODE_EXPANDABLE** + **AS_PRJITEM_MODE_SELECTABLE** + **AS_PRJITEM_MODE_NEWSUBFOLDERALLOWED**. 4. Value of the first property returned in the **ItemProperties** list by **ProjectWorkflowEvent_InitializeExpandItem**. 5. Value of the second property. 6. And so on. Your implementation of this event should set the constants in the third element above to meet your requirements.

Remarks

Not available in PowerWeb. This event and the **ProjectWorkflowEvent_InitializeExpandItem** and **ProjectWorkflowEvent_PrepareBrowser** events can be used to restrict the destination project folder selected by a user:

- The root folder to show
- Whether a subfolder is visible or not
- Whether a subfolder is selectable
- Whether a subfolder is expandable

After the event, all of the subfolders of the current folder are handled as indicated by the constants in the updated **SubItems** array. If an item does not occur in the **SubItems** array, the **AS_PRJITEM_MODE_NONE** constant is assumed.

Example

Following are example implementations of this event and the **ProjectWorkflowEvent_InitializeExpandItem** event:

```
'Rules for this example:
'1. Only 2nd level folders can be selected
'2. A folder is shown only when it is open and the current user's
```

```
name is on the list of users
'   stored in the CreatePCUsers property that are allowed to create a
project copy.
'3. Subfolders are not shown

Function ProjectWorkflowEvent_InitializeExpandItem(ItemProperties)
  lLevel = UBound (Split (Folder.Path, "\"))

  Select Case lLevel
    Case 2
      ItemProperties = Array ("ProjectControl.Open",
"ProjectControl.CreatePCUsers")
  End Select
End Function

Function ProjectWorkflowEvent_BeforeExpandItem(SubItems)
  lLevel = UBound (Split (Folder.Path, "\"))

  Select Case lLevel
    Case 0 'Root folders
      For i = LBound (SubItems) To UBound (SubItems)
        If Right (SubItems (1, i), 7) = "Projects"  Then
          SubItems (i)(2) = AS_PRJITEM_MODE_VISIBLE + AS_PRJITEM_
MODE_EXPANDABLE 'Show it
        Else
          SubItems (i)(2) = AS_PRJITEM_MODE_NONE 'Hide it
        End If
      Next
    Case 1 'Work Units
      For i = LBound (SubItems) To UBound (SubItems)
        SubItems (i)(2) = AS_PRJITEM_MODE_VISIBLE + AS_PRJITEM_MODE_
EXPANDABLE
      Next

     Case 2 'Splits
      For i = LBound (SubItems) To UBound (SubItems)

        If SubItems (i)(3) And InStr (User.Name & ";", SubItems (i)
(4)) > 0 Then
          'The split is available, the user can choose it
          SubItems (i)(2) = AS_PRJITEM_MODE_VISIBLE + AS_PRJITEM_
MODE_SELECTABLE
        Else
          SubItems (i)(2) = AS_PRJITEM_MODE_NONE 'Hide it
        End If
      Next
     Case Else
       'Cannot happen since everything at level 2 is not selectable
  End Select
End Function
```

InitializeExpandItem Event

Occurs before the folder selection dialog is shown to the user to select a destination folder for a project copy.

Syntax

```
ProjectWorkflowEvent_InitializeExpandItem (ItemProperties,
DialogTitle)
```

Parameters

Name	Description
ItemProperties	An array of fully qualified property names that should be set by your implementation of the event. This array will specify property filters that are supplied to the **ProjectWorkflowEvent_BeforeExpandItem** event.
DialogTitle	A string containing the text to be used for the folder selection dialog title.

Remarks

Not available in PowerWeb. This event and the **ProjectWorkflowEvent_BeforeExpandItem** and **ProjectWorkflowEvent_PrepareBrowser** events can be used to restrict the destination project folder selected by a user.

InitializeWizard Event

Occurs when a project workflow transition shows a property page wizard.

Syntax

```
ProjectWorkflowEvent_InitializeWizard (Batch, Transition)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Transition	A **WorkflowTransition** object that represents the transition that is showing the wizard. For more information about the **WorkflowTransition** object, see WorkflowTransition Object.

PrepareBrowser Event

Occurs before the folder selection dialog box is shown for the user to select a destination folder for a project copy.

Syntax

```
ProjectWorkflowEvent_ PrepareBrowser(RootFolder, DialogTitle,
ExpandableSublevelLimit)
```

Parameters

Name	Description
RootFolder	The path of the folder to show as the root in the **Select Folder** dialog.
DialogTitle	Custom title to show for the **Select Folder** dialog.
ExpandableSublevelLimit	Controls how many folder levels the user can expand in the **Select Project** dialog. Can be set to an integer value of **0** or higher. This argument is optional. If the argument is included in your configuration, the user also sees a message in the **Select Project** dialog which says, "Expanding the project folder is disabled by the configuration beyond level [X]".

Remarks

This event and the **ProjectWorkflowEvent_InitializeExpandItem** and **ProjectWorkflowEvent_BeforeExpandItem** events can be used to restrict the destination project folder selected by a user.

The folder selection dialog box can be shown for new non-project copy documents by setting `Client.Confirmation (AS_CONFIRM_NO_SELECTPROJECTWIZARD) = False` in the **DocGenericEvent_BeforeNewDocument** event. The selected folder can then be retrieved in the **DocGenericEvent_DocumentFolderSelected** event described in DocGenericEvent_DocumentFolderSelected event.

If the folder that was last selected by the user is a subfolder of the folder specified by **RootFolder**, it is selected by default in the folder selection dialog box. Otherwise, the folder specified by **RootFolder** is selected by default.

By default, when the project folder selection dialog box appears, the root of the vault is highlighted and can be selected. To prevent users from clicking the **OK** button and selecting the root and to force them to select or create a project folder, set the **AS_CONFIRM_ROOT_SELECTION** constant to **False** in this event. The default value is **True**.

This event does not occur in PowerWeb.

Examples

The following example shows how to initialize the **Select Folder** dialog box with a preset root folder. The user must then select a subfolder as the destination of the project copy. The user cannot select a project folder deeper than three levels in the folder structure.

```
Sub ProjectWorkflowEvent_PrepareBrowser(RootFolder, DialogTitle,
ExpandableSublevelLimit)
    RootFolder = "\projects"
    DialogTitle = "Select Project"
    ExpandableSublevelLimit = 3
End Sub
```

The following example demonstrates how to assign a project copy to a known destination project folder (retrieved from a document property, for example) and how to suppress the **Select Folder** and **Reference Explorer** (for reference selection if references exist) dialog boxes. The **DocCopyMoveEvent_PrepareCopy** event occurs before **ProjectWorkflowEvent_PrepareBrowser** and in this example, the browser is not shown.

```
Sub DocCopyMoveEvent_PrepareCopy(Batch, TargetFolder)
  If Not Document Is Nothing Then
    If Client.ImportType = AS_IT_COPIED And _
      Client.ImportDetails = AS_ID_CREATEPROJCOPY Then
      'Add your code for document objects
      TargetFolder = "\Design\Projects\894" 'Or some property value
      'Suppress folder selection dialog
      Client.Confirmation(AS_CONFIRM_PROJECT_FOLDER) = False
      'Suppress reference selection dialog
      Client.Confirmation(AS_CONFIRM_SELECT_ASSEMBLY_ITEMS) = False
    End If
  ElseIf Not Folder Is Nothing Then
    'Add your code for folder objects
  End If
End Sub
```

TerminateWizard Event

Occurs when a project workflow transition shows a property page wizard.

Syntax

```
ProjectWorkflowEvent_TerminateWizard (Batch, Transition)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Transition	A **WorkflowTransition** object that represents the transition that is showing the wizard. For more information about the **WorkflowTransition** object, see WorkflowTransition Object.

Remarks

Not available in PowerWeb.

Property Page Events

Property page events allow you to implement custom functionality for the following system-defined property pages:

- **Document**: The default property page for every document.
- **ExportPackages**: Shows the Meridian Explorer export packages related to the document.
- **Folder**: The default property page for every folder in the **Explorer** view.
- **ImportPackages**: Shows the Meridian Explorer import packages related to the document.
- **Objects**: Shows the asset tag objects related to the document.
- **Rendition**: Shows the properties in the **BCRenditionPropertySet** property set.
- **Retention**: Shows the properties in the **AMRetentionControlPropertySet** property set.
- **TitleBlocks**: Shows the values of title block attributes when multiple layouts exist in a drawing and the **Synchronize the title blocks in all layouts** option is enabled as described in *Configure Title Block Updates* in the *Meridian Enterprise Configuration Guide*.
- **WhereUsed** Shows the documents related to the current asset object.

System Property Page Event Sequence

The events that occur for the system property pages are shown in the following list in the sequence that they occur.

- *<PageName>*Page_IsVisible
- *<PageName>*Page_CanEdit
- *<PageName>*Page_CanApply(Context)

<PageName>Page_IsVisible Event

Occurs before one of the pages listed is shown in the Meridian client applications.

Syntax

```
<PageName>Page_IsVisible As Boolean
```

Remarks

Return **False** (default) to hide the property page.

Example

```
Function ExportPackagesPage_IsVisible()
    If Not Document Is Nothing Then
        'Add your code for document objects
        ExportPackagesPage_IsVisible = True
    ElseIf Not Folder Is Nothing Then
        'Add your code for folder objects
    End If
End Function
```

<PageName>Page_CanEdit Event

Controls whether the **Edit** button appears in the Meridian client applications for one of the pages listed except for the **Document**, **Folder**, and **Import Packages** pages (not editable).

Syntax

```
<PageName>Page_CanEdit As Boolean
```

Remarks

Return **False** to prevent users from editing the property page. The name of the event for the **Title Blocks** page is `TitleBlocksPage_CanEdit` (no spaces).

<PageName>Page_CanApply Event

Occurs when a user clicks **Apply** or **Close** in the Meridian client applications for one of the pages listed except for the **Document**, **Folder**, and **Import Packages** pages (not editable). This event occurs once for every changed property value.

If this event returns **False**, the corresponding grid cell on the **Title Blocks** property page is switched to edit mode and will wait for the user to modify the current value.

Syntax

```
<PageName>Page_CanApply(Context) As Boolean
```

Parameters

Name	Description
Context	A one dimension array that contains the layout name, block name, property name, old value, and new value as shown in the example. Read only.

Remarks

If you use this event for the **Retention** or **Rendition** property pages, you do not need to show your own dialog box (**WinMsgBox** function) for validation errors. Instead, you can raise an error as in the following example, and it will be shown by the property page accordingly.

```
err.Raise vbObjectError, "RetentionPage_CanApply", "Invalid value for EXPIRATIONDATE"
```

You can also use this event to validate the property values that are mapped to attributes in the title blocks in multiple layouts of a drawing. The name of the event for the **Title Blocks** page is `TitleBlocksPage_CanApply` (no spaces). The DocGenericEvent_OnProperties event occurs after this event.

Example

```
Function TitleBlocksPage_CanApply(Context)
    If Not Document Is Nothing Then
        If IsArray(Context) Then
            Dim s
            s = "Layout = " + CStr(Context(0)) + ", "
            s = s + "Block = " + CStr(Context(1)) + ", "
            s = s + "Property = " + CStr(Context(2)) + ", "
            s = s + "Old value = " + CStr(Context(3)) + ", "
            s = s + "New value = " + CStr(Context(4))
            If UCase(CStr(Context(4))) = LCase(CStr(Context(4))) Then
                TitleBlocksPage_CanApply = True
                s = s + ": VALID"
            Else
                TitleBlocksPage_CanApply = False
                s = s + ": INVALID"
            End If
            WinMsgBox s
        End If
    End If
End Function
```

Publishing Events

Publishing events occur in Meridian Enterprise vaults when they are configured as the source or destination document management system.

DocSync_Failed Event

Occurs for the source document after synchronization fails.

Syntax

```
Function DocSync_Failed()
```

Remarks

Use this event with the **Enable separate options per document** option and the **Enable Meridian script events** option of a repository synchronization job to control synchronization per document.

DocSync_Options Event

Occurs for the source document before synchronization starts.

Syntax

```
Function DocSync_Options()
```

Remarks

Use this event with the **Enable separate options per document** option and the **Enable Meridian script events** option of a repository synchronization job to control synchronization per document.

DocSync_Succeeded Event

Occurs for the source document after synchronization succeeds.

Syntax

```
Function DocSync_Succeeded()
```

Remarks

Use this event with the **Enable separate options per document** option and the **Enable Meridian script events** option of a repository synchronization job to control synchronization per document.

PublisherDestinationEvent_AfterPublish Event

Occurs in the destination vault after publishing has ended.

Syntax

```
PublisherDestinationEvent_AfterPublish (sourceType, sourceAddress, _
    sourceDocId, publishOptions)
```

Parameters

Parameter	Description
sourceType	The name of the source system link, for example, **BC Meridian** or **Windows file system**.
sourceAddress	The address of the source system, for example, **DataStoreName@MachineName** for the AccruentMeridian Enterprise link.
sourceDocId	The ID of document in the source system.
publishOptions	The publishing options that were applied to the job. If the options were specified in the source system client application, they are passed in this parameter. For information about the publishing options supported by a specific system link, see the system link description in the *Publishing Modules* section in the *Meridian Enterprise Server Administrator's Guide*. If rendering fails with any rendering module and another attempt is configured, this parameter contains the text **_RETRYRENDER_*<ModuleName>*_** . Your implementation of this event can then perform other actions in response to the failure, such as notify a System Administrator, set a property value for easy document location, and so on.

Example

```
Public Sub PublisherDestinationEvent_AfterPublish(sourceType,
sourceAddress, _
    sourceDocId, publishOptions)
    Document.Log _
       "Published: " & _
       "From " & sourceType & " '" & sourceAddress & "'" & _
```

```
        " document with ID '" & sourceDocId & "'" & _
        " with options '" & publishOptions & "'"
End Sub
```

PublisherDestinationEvent_HotspotUpdated Event

Occurs in the destination vault after publishing has ended if the Hotspot Extraction option **Publisher will invoke Meridian script event when update succeeds** is selected. For more information about this option, see the *Configure Hotspot Extraction Options* article in the *Meridian Enterprise Server Administrator's Guide*.

Syntax

```
PublisherDestinationEvent_HotspotsUpdated ()
                        CreateTags (Document.SaveToFile)
```

PublisherSourceEvent_BeforePublish Event

Occurs in the source vault before publishing starts.

Syntax

```
PublisherSourceEvent_BeforePublish (destType, destAddress, _
    destDocId, destDocPath, destDocName, publishOptions)
```

Parameters

Parameter	Description
destType	The name of the destination system link, for example, **BC Meridian** or **Windows file system**.
destAddress	The address of the destination system as shown on the **Destination** page, for example, **DataStoreName@MachineName** for the Meridian Enterprise system link.
destDocId	The ID of the document in the destination system. Contains an empty value if the document does not yet exist in the destination system.
destDocPath	The path of the document in the destination system.
destDocName	The name of the document in the destination system.

Parameter	Description
publishOptions	The publishing options to apply to the job. If the options were specified in the source system client application, they are passed in this parameter. For information about the publishing options supported by a specific system link, see the system link description in the *Publishing Modules* section in the *Meridian Enterprise Server Administrator's Guide*.

Return Value

An array of variants. The first element in the array is treated as a new value for **DestDocID** parameter.

Example

```
Function PublisherSourceEvent_BeforePublish (destType, destAddress, _
    destDocId, destDocPath, destDocName, publishOptions)
    Document.Log _
       "Started publishing: " & _
       "to " & destType & " '" & destAddress & "'" & _
       " document with ID '" & destDocId & "'" & _
       " with options '" & publishOptions & "'" & _
       " and path '" & destDocPath & "' and name '" & destDocName & "'"
End Function
```

PublisherSourceEvent_AfterPublish Event

Occurs in the source vault after publishing has ended.

Syntax

```
PublisherSourceEvent_AfterPublish (destType, destAddress, _
    destDocId, publishOptions)
```

Parameters

Parameter	Description
destType	The name of the destination system link, for example, **BC Meridian** or **Windows file system**.

Parameter	Description
destAddress	The address of the destination system as shown on the **Destination** page, for example, **DataStoreName@MachineName** for the AccruentMeridian Enterprise link.
destDocId	The ID of document in the destination system. Contains an empty value if the document does not yet exist in the destination system.
publishOptions	The publishing options to apply to the job. If the options were specified in the source system client application, they are passed in this parameter. For information about the publishing options supported by a specific system link, see the system link description in the *Publishing Modules* section in the *Meridian Enterprise Server Administrator's Guide*. If rendering fails with any rendering module and another attempt is configured, this parameter contains the text **_RETRYRENDER_<ModuleName>_** . Your implementation of this event can then perform other actions in response to the failure, such as to notify a System Administrator, set a property value for easy document location, and so on.

Example

```
Public Sub PublisherSourceEvent_AfterPublish (destType, destAddress,
_
    destDocId, publishOptions)
    Document.Log _
       "Published: " & _
       "to " & destType & " '" & destAddress & "'" & _
       " document with ID '" & destDocId & "'" & _
       " with options '" & publishOptions & "'"
End Sub
```

Vault Events

The vault events occur when actions affect the entire vault. These events do not occur in the PowerWeb.

Vault Event Sequences

The events that occur for the vault commands are shown in the following lists in the sequence that they occur.

Change scope event sequence

- **VaultEvent_ChangeScope**

Change view event sequence

- **VaultEvent_ChangeView**

Close event sequence

- **VaultEvent_Close**
- **VaultEvent_ChangeView**

Open event sequence

- **VaultEvent_Open**

ChangeScope Event

This event can be invoked by script to present the user with a list of scopes from which to choose. The scopes can be existing ones defined in the vault configuration or this procedure can define a temporary virtual scope based upon an existing one like a template.

Syntax

`VaultEvent_ChangeView(`*`ScopeName`*`, `*`Scope`*`)`

Parameters

Name	Description
ScopeName	A string that contains the name of an existing scope to set as the default for the user or the name of a virtual scope that the procedure will create based on an existing scope. No special characters except the underscore (_) that appear in the scope list shown to the user (for example, a hyphen) will be returned in this property name (the user selected scope). The procedure itself must parse and replace such characters.
Scope	The scope object set for the user by the procedure. Its value is `Nothing` when the procedure is first called. References to this object before it has been initialized by the procedure will produce the error **Cannot initialize user session**.

Remarks

This procedure is intended to be used in vaults that contain many projects and by users who should be limited by scopes to only work on those projects to which they are assigned. Scopes defined in Meridian Configurator cannot be modified by this procedure. The virtual scopes defined by this procedure exist only within the current vault transaction. This procedure is not supported in PowerUser.

The `DynamicScopes` property used in this procedure is a one dimension array of the internal names of scopes to present to the user for selection. The names must not contain any leading numbers or non-English characters. The names should not contain any spaces until after the scope has been defined, then its **DisplayName** property can be set and include spaces. The array replaces the normal list of scopes defined in the vault configuration. All scope names in this array must be handled by this procedure. If the user selects an unhandled virtual scope, no scope will be applied and errors can occur. No automatic security validation is performed, it is the responsibility of the procedure to do so, if required.

Example

```
Sub VaultEvent_ChangeScope(ScopeName,Scope)
  ' Set scope based on user name
  If (User.Name = "Benjamin") Then
    ' Create a scope based on an existing scope named Manager
    Set Scope = Vault.Scope("Manager")
    ' Give it the project name
    Scope.DisplayName = ScopeName
    ' Set its root folder based on the project name
    If (ScopeName = "Project123") Then
      Scope.RootFolder = "\West\Metropolis\123"
    ElseIf (ScopeName = "Project234") Then
      Scope.RootFolder = "\West\Metropolis\234"
```

```
    ElseIf (ScopeName = "Project345") Then
      Scope.RootFolder = "\West\Metropolis\345"
    Else
      ' Other projects use a default scope
      Set Scope = Vault.Scope(ScopeName)
    End If
    ' Show available scopes for Benjamin
    Scope.DynamicScopes = Array ("Project 123", "Project 234",
"Project 345")
  Else
    ' Other users get the default scope
    Set Scope = Vault.Scope(ScopeName)
  End If
End Sub
```

ChangeView Event

Occurs when a user selects a different navigation view of the vault.

Syntax

`VaultEvent_ChangeView(`*View*`)`

Parameters

Name	Description
View	A string that contains the name of the view selected by the user.

Remarks

Not available in PowerWeb.

Close Event

Occurs when a user closes a vault.

Syntax

`VaultEvent_Close()`

Parameters

This event receives no parameters.

Remarks

Not available in PowerWeb.

NewProfile Event

Occurs after a new profile file is created for a user and they open a vault for the first time in PowerWeb.

Syntax

```
VaultEvent_Open()
```

Parameters

This event receives no parameters.

Open Event

Occurs when a user opens a vault.

Syntax

```
VaultEvent_Open()
```

Parameters

This event receives no parameters.

Remarks

Not available in PowerWeb. This event is not invoked by the Document Import tool. Consider using the Option property of the **Vault** object with values stored on the **Settings** page in the **Vault Settings** group in the **Environment** branch in Meridian Enterprise Configurator instead.

Workflow Definition Events

The events in this category occur when a workflow definition is active.

Workflow Event Sequences

The events that occur for the workflow definition commands are shown in the following lists in the sequence that they occur.

Execute transition event sequence

- **DocGenericEvent_PrepareCommand**: :Occurs in PowerUser only.
- **DocCWFEvent_PreInitializeExecuteTransition**: Occurs in PowerWeb only.
- **DocCWFEvent_InitializeExecuteTransition**
- **DocCWFEvent_InitializeWizard**
- **DocGenericEvent_OnProperties**
- **DocCWFEvent_TerminateWizard**
- **DocCWFEvent_PreBeforeExecuteTransition**
- **DocCWFEvent_BeforeExecuteTransition**
- **DocCWFEvent_BeforeNewRevision**
- **DocCWFEvent_AfterNewRevision**
- **DocCWFEvent_TerminateNewRevision**
- **DocCWFEvent_AfterExecuteTransition**
- **DocWorkflowEvent_AfterReplaceReleased**
- **DocCWFEvent_TerminateExecuteTransition**

Reassign Managers event sequence

- **DocGenericEvent_PrepareCommand**
- **DocWorkflowEvent_InitializeChangeWFManager**: Occurs in PowerUser only.
- **DocWorkflowEvent_BeforeChangeWFManager**
- **DocWorkflowEvent_AfterChangeWFManager**
- **DocWorkflowEvent_TerminateChangeWFManager**: Occurs in PowerUser only.

Reroute event sequence

- **DocGenericEvent_PrepareCommand**
- **DocCWFEvent_InitializeReroute**
- **DocCWFEvent_BeforeReroute**
- **DocCWFEvent_AfterReroute**
- **DocCWFEvent_TerminateReroute**

*ExecuteTransition Events

Occurs when a workflow definition transition is executed.

The new version of the document is created between the **BeforeExecuteTransition** and the **AfterExecuteTransition** event. For the first transition in the workflow, this means that in the **BeforeExecuteTransition** event and all preceding events, the **Document** object in script refers to the released revision of the document. In the **AfterExecuteTransition** event and subsequent events, the **Document** object refers to the new revision in the workflow.

As a result, all changes made to the **Document** object (to property values, for example) in script in the events up to and including the **BeforeExecuteTransition** event will change the released revision and will not be revoked if the workflow is revoked.

To prevent unexpected changes to the released document, we recommend avoiding making changes to the **Document** object in the events up to and including the **BeforeExecuteTransition** event.

Syntax

`DocCWFEvent_*ExecuteTransition(`*`Batch`*`, `*`Transition`*`, `*`Person`*`, `*`Manager`*`, `*`Comment`*`)`

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Transition	A WorkflowTransition object that represents the transition to execute.
Person	An object that represents the user who is executing the transition.
Manager	An object that represents the user who is the manager of the workflow.
Comment	A string containing the text entered by the user who is executing the transition.

Remarks

This set of events is intended to support user input to a batch process using the Batch.Confirmation Method and its related methods and properties as follows:

- The user initiates a workflow transition on a batch of documents
- **DocCWFEvent_PreInitializeExecuteTransition** event is executed and any changes to the event parameters are ignored. Not supported by PowerUser.

  Any batch scope confirmations are shown at this time.
- **DocCWFEvent_InitializeExecuteTransition** is executed and any changes to the event parameters are processed.

  Any batch scope confirmation results can be queried in script.
- For each document in the batch:
  - **DocCWFEvent_PreBeforeExecuteTransition** is executed and any changes to the event parameters are ignored. Not supported by PowerUser.

    Any document scope confirmations are shown at this time. The document name is visible to the user.
  - **DocCWFEvent_BeforeExecuteTransition** is executed and any changes to the event parameters are processed.

    Any document scope confirmation results can be queried in script.
  - **DocCWFEvent_AfterExecuteTransition** is executed and any changes to the event parameters are ignored.
  - **DocCWFEvent_TerminateExecuteTransition** is executed and any changes to the event parameters are ignored.

*NewRevision Events

Occurs when a new revision is created by a transition of the workflow.

Syntax

```
DocCWFEvent_*NewRevision(Batch, Transition)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.

Name	Description
Transition	A **WorkflowTransition** object that represents the transition that is creating the new revision.

*Reroute Events

Occurs when a workflow is rerouted to a state other than a default destination state.

Syntax

```
DocCWFEvent_*Reroute(Batch, State, Person, Comment)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
State	An object that represents the destination state.
Person	An object that represents the user who is rerouting the documents.
Comment	A string containing the text entered by the user who is rerouting the documents.

*Wizard Events

Occurs when property pages are shown by a transition of the workflow.

Syntax

```
DocCWFEvent_*Wizard(Batch, Transition, Comment)
```

Parameters

Name	Description
Batch	An object that represents the batch of documents the user has selected.
Transition	A **WorkflowTransition** object that represents the transition that is showing the property pages.
Comment	A string that represents the default comment to add to the comment log.

Meridian API Constants

The Meridian API provides many constants. A constant is a meaningful name that represents a number or a string and never changes. You can use constants to query for and specify application states and options. These constants can be used with their related methods (actions) and properties (data) to manipulate vault objects and control application behavior. The Meridian constants are members of enumerations that are each used for a specific purpose. You can view the constants that are contained in each enumeration with the **Object Browser** of the Meridian Enterprise Script Editor. The tooltip text of each constant gives a brief description of the constant.

The following table describes each Meridian constant enumeration.

Meridian API constant enumerations

Enumeration	Description
AS_BRC_IMPORT_ ACTION	Briefcase import actions
AS_CALLREMOTE_ FLAGSS	Remote procedure call types
AS_CE_RULE	Enumerations of behaviors that you can specify for a project copy: • `AS_CER_DEFAULT`: Default behavior • `AS_CER_MERGE_WF`: Current behavior • `AS_CER_NOT_ALLOWED`: Allow only one project copy • `AS_CER_SERIAL_WITH_RELEASE`: Allow multiple project copies • `AS_CER_SERIAL_WITH_TRANSFER`: Reserved for future use
AS_CI	Table column information index
AS_CLIENTID	Meridian client IDs
AS_CMD_MODE	Custom command modes
AS_CMD_STATE	Custom command visibility states
AS_CONFIRM_ ACTION	Confirmation prompt toggles
AS_FEATURES	Advanced and module features supported by the vault.
AS_GROUP_ COLUMNS	Meridian group information column sets

Enumeration	Description
AS_HOTSPOTS_TYPE	Enumerations that you can use to specify whether a method is carried out for a particular type of hotspot on a document: • `AS_HOTSPOTS_AUTOMATIC`: Automatically generated hotspots • `AS_HOTSPOTS_MANUAL`: Manually created hotspots
AS_HYBRID_ACTION	Hybrid document part actions
AS_IMPORTDETAILS	Document import details
AS_IMPORTTYPE	Document import actions
AS_INCLUDE_ CONTENT_OPTIONS	What content should be included: the rendition, the source document, or both.
AS_LOGFLAGS	Event logging parameters
AS_MAPIMSG_ RECIP_TYPE	MAPI message recipient types
AS_MAPIMSG_ SEND_FLAGS	MAPI message initiation types
AS_MOVE_OPTIONS	Document move options
AS_MsgBoxResult	Message box result codes
AS_MsgBoxStyle	Message box style options
AS_NEWFOLDER_ OPTIONS	Folder creation options
AS_PCLOCK	Project copy lock types
AS_PORTAL_ OPTIONS	Allows you to choose between sending the package for revision or for information. Also allows you to send the latest released revision of the documents instead of the working copy. To learn more about these options, see *Send Documents To Meridian Portal* in the *Meridian Enterprise User's Guide*.
AS_PRIVILEGES	Security privileges
AS_PROJITEM_ FLAGS	Project selection dialog configuration options
AS_PROP_CMD	Property page commands
AS_PWF_STATUS	Project folder workflow status

Enumeration	Description
AS_REFRESHFLAG	Client refresh options
AS_SUBMITTAL_STATUS_FLAGS	Submittal states. Used by Meridian Transmittal Management Module.
AS_SUBMITTAL_STATUS_VALUES	Current submittal status. Used by Meridian Transmittal Management Module.
AS_TQTYPE	External data table or query types
AS_TRANSMITTAL_STATUS_FLAGS	Transmittal states. Used by Meridian Transmittal Management Module.
AS_TRANSMITTAL_STATUS_VALUES	Current Transmittal status. Used by Meridian Transmittal Management Module.
AS_URL_FLAGS	PowerWeb URL formats
AS_USER_COLUMNS	Meridian user information column sets
AS_WA_STATEFLAG	Work area statuses
AS_WATERMARK_COLORS	Watermark colors
AS_WATERMARK_STYLES	Watermark annotation style options
AS_WATERMARK_TITLE_STYLES	Watermark title style options
AS_WF_STATE	Workflow definition current status
AS_WFINTERLOCK_LOCATION	Workflow definition interlock level
AS_WORKFLOW_STATE_TYPE	Workflow definition state types
AS_WORKFLOW_TRANS_RES	Expected result of executing a transition
IC_OPERATOR	Search criteria operators
IC_SHOWWINDOW	Shell command window options

VBScript Examples

The following topics describe how to use VBScript to automate common tasks. You can use these examples in your own vault configuration. For expert assistance with VBScript customization, contact the Accruent Professional Services department or a Solution Partner.

Validating unique property values

Nearly every vault has at least one property that must only contain unique values. That is, the value should not be duplicated anywhere in the vault, for example, document names or numbers. Just like Windows, Meridian prevents duplicate document names in the same folder, but to prevent duplicates in an entire vault requires a VBScript validation expression.

The **IsUniqueValue** method of the **Document** object queries the vault for a given value in a given property. You can use this method in an expression for the **This expression must Be True** option on the **Validation** page of a custom property:

```
Document.IsUniqueValue (Document.MyProperty, Value)
```

Clearing the property values of copied documents

When documents are copied, their property values are copied as well. This may be undesirable for some properties that you want the user to enter or that should be unique. To accomplish this, you need VBScript to react and clear the properties when documents are copied. A custom event procedure for the **DocGenericEvent_BeforeNewDocument** event is the ideal solution. This event occurs just before document copies are created, when you can set the property values that will be saved for the copies. If the properties are not used for all document types in the same vault, you can use a conditional statement to take the appropriate action for each document type as in the following example.

```
Sub DocGenericEvent_BeforeNewDocument(Batch, Action, SourceFile,
DocType, DocTemplate)
    If Document.DocumentType.InternalName = "QualityDocument" Then
        Document.Category = "Quality Documents"
        Document.LastAuditBy = vbNullString
        Document.LastAuditOn = Null
        Document.AuditResult = vbNullString
        Document.AuditorNotes = vbNullString
        Document.NextAuditBy = vbNullString
        Document.NextAuditOn = Null
        Document.AuditHistory = vbNullString
        Document.Author = vbNullString
        Document.ApprovedBy = vbNullString
        Document.ApprovedOn = Null
```

```
        Document.Comments = vbNullString
    Else If Document.DocumentType.InternalName = "NonConformance"
Then
        Document.Category = "Non Conformances"
    End If
End Sub
```

> **Note:**
> Date properties do not accept string values. Therefore, instead of setting them to **vbNullString**, set them to **Null**.

Setting property values based on workflow transitions

It can be convenient for some property values to be set automatically at specific transitions in a document's workflow. This can be done by determining the state to which a document is being routed and setting property values accordingly. Meridian provides many constants that you can use, including the **AS_WF_STATE** enumeration used in the following example.

```
Sub DocWorkflowEvent_AfterChangeWFState(Batch, SourceState,
TargetState, Person, Comment)
    If TargetState = AS_WF_APPROVED Then
       Document.ApprovedOn = vbDate
       Document.ApprovedBy = User.FullName
    End If
    If TargetState = AS_WF_UNDERCHANGE Then
       Document.ApprovedOn = Null
       Document.ApprovedBy = ""
    End If
End Sub
```

Sending email messages

Automatically generating email messages is useful for many purposes, particularly during customized event procedures. The following example code shows a typical procedure for setting properties of a **NewMailMessage** object. It uses constants in the **AS_MAPIMSG_RECIP_TYPE** and **AS_MAPIMSG_SEND_FLAGS** enumerations to set various properties of the message before sending it.

```
Sub SendEmailMessage()
    Dim oMessage
    Set oMessage = Client.NewMailMessage

    'Recipients
    oMessage.Recipients.Add "",AS_MMRT_
TO,"<recipient.name@anotherdomain.com>"
    oMessage.Recipients.Add "",AS_MMRT_
```

```
CC,"<recipient.name@anotherdomain.com>"

    'Sender
    oMessage.Originator.Address = "<sender.name@mydomain.com>"
    oMessage.Originator.Name = "Sender's Name"
    oMessage.Originator.Type = AS_MMRT_ORIG

    'Subject
    oMessage.Subject = "The subject of the message"

    'Body text
    oMessage.NoteText = "The body text of the message"

    'Attachments
     oMessage.Attachments.Add "c:\temp\my.bmp", ""

    'Send it with MAPI options and reset when done
    oMessage.Send AS_MMSF_LOGON_UI
    oMessage.Clean
End Sub
```

Automation Objects

Automation objects are special objects that applications (called *automation servers*) make available to other applications. The objects make it possible for functionality or data within one application to be used inside the other application, for example, a spreadsheet within a word processing document. This is convenient for application users because they do not need to leave one application in order to use the other application.

Automation servers provide at least one type of object. For example, a word processing application might provide an application object, a document object, and a toolbar object. Automation objects can be created with the Windows client applications of Meridian with VBScript. To create an automation object, assign the object returned by **CreateObject** to an object variable as in the following example:

```
Dim ExcelSheet
Set ExcelSheet = CreateObject("Excel.Sheet")
```

This code starts the application and creates the object (in this example, a Microsoft Excel worksheet). After an object is created, refer to it in your code using the object variable you defined. As shown in the following example, you can access properties and methods of the new object using the object variable created above, **ExcelSheet**, and other Excel objects, including the **Application** object and the **ActiveSheet.Cells** collection:

```
' Make Excel visible through the Application object.
ExcelSheet.Application.Visible = True
' Place some text in the first cell of the sheet.
ExcelSheet.ActiveSheet.Cells(1,1).Value = "This is column A, row 1"
' Save the sheet.
ExcelSheet.SaveAs "C:\DOCS\TEST.XLS"
' Close Excel with the Quit method of the Application object.
ExcelSheet.Application.Quit
' Release the object variable.
Set ExcelSheet = Nothing
```

By default, automation objects are created on the computer where the code is executed. You can create an object on a remote networked computer by passing the name of the computer to the **ServerName** argument of **CreateObject**. In addition, you can specify **ServerName** using DNS format or an IP address. The following code returns the version number of Excel running on a remote network computer named **MyServer**:

```
Function GetVersion

Dim XLApp

    Set XLApp = CreateObject("Excel.Application", "MyServer")
    GetVersion = XLApp.Version
```

```
End Function
```

An error occurs if the specified remote computer does not exist or cannot be found, for example, because of security restrictions.

> **Note:**
> **CreateObject** is suitable for creating automation objects that do not act upon Meridian objects. For access to Meridian objects, use the AMCreateObject Function instead, which works the same as **CreateObject** but also passes the current **Repository** object to the **IAMCommandSetInitialize** interface of the automation server.

The following example defines a procedure for a custom command that creates a **Connection** object to connect to an Access database, creates a **Recordset** object to hold data from the connection, and sets a document property to a value from the **Recordset** object:

```
Sub ReadSettings_Execute(Batch)

    Dim cn, rs

    Set cn = CreateObject ("ADODB.Connection")
    Set rs = CreateObject ("ADODB.Recordset")

    cn.ConnectionString = _
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\temp\test.mdb;Persist Security Info=False"

    cn.Open
    rs.Open "SELECT * FROM Address", cn

    Document.Test = rs.Fields(1).Value

    rs.Close
    cn.Close
    Set rs = Nothing
    Set cn = Nothing

End Sub
```

Object Arguments

An *argument* is an object property that you create to hold a value. Arguments can be used in VBScript as global variables. An argument can contain any value, but it should not hold a reference to an object. An argument can have any name. Meridian supports arguments for the **Vault**, **Client**, and **Batch** objects. Object arguments are case-sensitive.

Following is an example of obtaining email information from **Client** arguments:

```
Sub DocGenericEvent_AfterNewDocument(Batch, Action, SourceFile, DocType, DocTemplate)
    vSubject = Client.Argument("Subject")
    vSenderName = Client.Argument("From")
        If IsEmpty(vSenderName) Then
            vSenderName = Client.Argument("SenderName")
        End If
        WinMsgBox("AfterNewDocument( " & Document.FileName & _
                " ): EmailSubject= [ " & vSubject & _
                " ], SenderName= [ " & vSenderName & " ]")
End Sub
```

Some objects provide predefined arguments that are set with values by Meridian during certain commands. Although these predefined arguments may contain object references, they are validated internally in the context of their intended uses. Arguments that you create are not validated and errors can occur if they are set to objects.

Only string values are supported. To store values of other types, the values need to be converted to and from string.

Numeric Example

```
Client.Argument (“Num”) = MyNumber 'converted automatically
MyNumber = Val (Client.Argument (“Num”)) 'convert back to number
```

Boolean Example

```
Client.Argument (“Bool”) = IIf (MyBool, “1”, “0”)
MyBool = Client.Argument (“Bool”) = “1”
```

VBScript and PowerWeb

Not all VBScript functions are available to PowerWeb since it runs on the Meridian web server, not on the client computer.

VBScript Functions Not Available in PowerWeb

Following are the major functions that are not available, with corresponding recommended workarounds.

VBScript functions not available in PowerWeb

Function	Workaround
Global variables	Use object arguments instead as described in Object Arguments.
DebugAssert Function	Not supported
WinInputBox Function	AskInput
WinMsgBox Function	AskConfirmation
ProjectWorkflowEvent_InitializeExpandItem and **ProjectWorkflowEvent_BeforeExpandItem** events	Not supported
Confirmation property The Client object methods CurrentView property	Not yet available
Vault Events	None
AddDocumentToBatch method and AddDocuments method	Not yet available
	Import packages
DocGenericEvent_PrepareCommand event	Not yet available
DocGenericEvent_OnProperties event when creating new folders	Not yet available
DocGenericEvent_AfterPrint event	Not yet available
AS_PROP_CMD constants **AS_PS_CMD_VIEW** and **AS_PS_CMD_CANCEL** in the DocGenericEvent_OnProperties event	Assign property validation conditions or use **AS_PS_CMD_APPLY** instead.
AS_NFO_SHOWWIZARD constant	Not yet available.

Notes About Functionality

- The values of check boxes and radio buttons on PowerWeb pages are sent to the web server only if they are selected (**True**). Meridian Enterprise Boolean properties that have not been explicitly set to either **True** or **False** contain no value. For these reasons, we recommend that you always test the values of Boolean properties for **True** and never for **False** whether the script is for PowerWeb or not.
- A script that is not compatible with PowerWeb will not generate an error message. Thoroughly test all VBScript configuration expressions and event procedures that might be invoked by PowerWeb.
- We highly discourage programmatic formatting of dates for use in title blocks, especially if PowerWeb will be used. Dates are transferred over the Internet in a locale-independent format to accommodate the possibility that the web server and PowerWeb locales are different. We recommend that you use the date attribute formatting settings described in the *Format attribute date values* article in the *Meridian Enterprise Configuration Guide* instead.

Formatting Text With RTF Codes

You can format the text shown by the **WinInputBox** and **WinMsgBox** functions using a limited set of Rich Text Format (RTF) codes within the text. Many of the codes use the plus symbol (+) and minus symbol (-) to enclose the text that they affect. The codes are only evaluated for the **Prompt** argument of the functions and do not work in the **Title** argument text.

The following table describes the available codes.

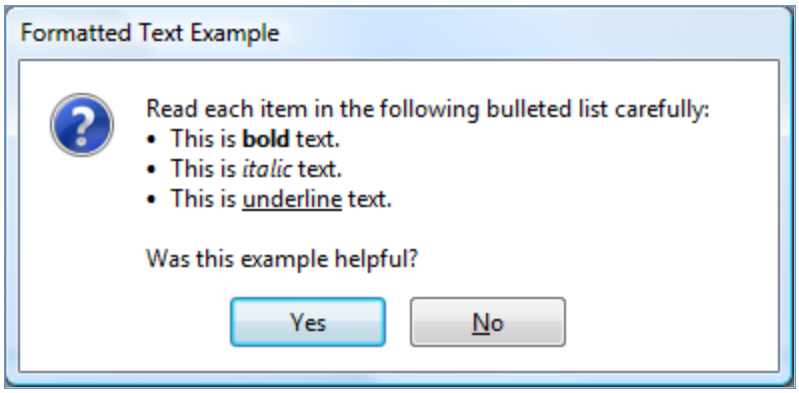
Text formatting codes

Code	Description
$RTF+/-	Enable or disable formatting
$NL	Inserts a new line
$CENTER+	Center justification
$RIGHT+	Right justification
$RESET	Reset text justification (left justification)
$NUM+/-	Numbered list
$B+/-	Bold attribute
$I+/-	Italic attribute
$U+/-	Underline attribute
$UT+/-	Thick underline attribute
$UDOT+/-	Dot underline attribute
$UD+/-	Dash underline attribute
$UDD+/-	Dot dash underline attribute
$UDDD+/-	Dot dot dash underline attribute
$UW+/-	Wavey underline attribute
$STR+/-	Strikethrough attribute
$SUB+/-	Subscript attribute
$SUP+/-	Superscript attribute
$CAPS+/-	All capitals attribute

Code	Description
$BULLET+/-	Bullet list
$BLACK+/-	Black color attribute
$BLUE+/-	Blue color attribute
$CYAN+/-	Cyan color attribute
$GREEN+/-	Green color attribute
$MAGENTA+/-	Magenta color attribute
$RED+/-	Red color attribute
$YELLOW+/-	Yellow color attribute
$WHITE+/-	White color attribute
$DBLUE+/-	Dark blue color attribute
$DCYAN+/-	Dark cyan color attribute
$DGREEN+/-	Dark green color attribute
$DMAGENTA+/-	Dark magenta color attribute
$DRED+/-	Dark red color attribute
$DYELLOW+/-	Dark yellow color attribute
$DGRAY+/-	Dark gray color attribute
$GRAY+/-	Gray color attribute
$AUTO	Automatic color attribute
$B_<*ColorName*>+/-	Background color attribute where <*ColorName*> is one of the colors listed above.

The following example VBScript code incorporates several formatting codes to produce the result shown in the following figure.

```
WinMsgBox "$RTF+Read each item in the following bulleted list
carefully:$NL" & _
          "$BULLET+This is $B+bold$B- text.$NL" & _
          "This is $I+italic$I- text.$NL" & _
          "This is $U+underlined$U- text.$NL" & _
          $BULLET-$NLWas this example helpful?$RTF-", _
          AS_QUESTION Or AS_YESNO, _
          "Formatted Text Example"
```


Formatted Text Example
Read each item in the following bulleted list carefully:
• This is bold text.
• This is italic text.
• This is underline text.
Was this example helpful?
Yes
No

Confirmation Pages

In 2020 R2, we added VBScript support for implementing custom confirmation pages. Confirmation pages are custom dialog boxes which are triggered at certain points of a workflow, as part of a wizard, or as part of a command. There are multiple events that can be used to trigger a confirmation page. See examples of how confirmation pages can be implemented in our KnowledgeBase.

This topic describes the purpose of confirmation pages, their benefits and limitations, and provides a simplified script sequence to demonstrate where in a script confirmation pages can be triggered. In this topic you can also find the VBScript events that support confirmation pages, the properties and methods you can use to build confirmation pages, and how to add confirmation pages to your VBScript.

Learn more about creating and editing event procedures.

Background

When a user executes an operation in Meridian, it may be necessary to gather input or show information about the item being processed. What input is needed or what information should be shown can depend on the metadata of the item, and complex logic can be involved in making the decision.

In PowerWeb, the Meridian script is executed on the web server. Due to the nature of web applications, it is not possible to trigger a window to display in the browser from a script executing on the server. To address this, the Confirmation Page functionality has been added as an alternative to the **WinMsgBox** and **WinInputBox** functions.

Confirmation pages offer more user interface customization options than **WinMsgBox** and **WinInputBox**. There are more options for input fields and multiple inputs can be combined in a single page. This simplifies the data collection process.

However, there are limitations to when confirmation pages can be triggered. Displaying a window and gathering user input requires a round trip between the server and the browser. To initiate this process, additional events with the prefix **Pre**have been added. You can use the handlers for these events to control the display of input fields.

After processing the script, the browser will show a confirmation page. The user will view the information and enter the required data. Their input is then posted back to the server and can be read using script functions in the next event.

Simplified Script Sequence

The numbered list below describes the general sequence of events for confirmation pages in VBScript. This sequence can vary depending on the event type, and whether the confirmation page is part of a wizard.

The general sequence is:

1. The user selects a batch of items.
2. The user executes a command.
3. The **PreInitialize** event fires.
    - This occurs once for the entire batch.
    - Script logic can be added to this event to show information or gather input applicable to the entire batch.
4. The Confirmation Page for batch initialization is shown.
5. The **Initialize** event fires.
    - This occurs once for the entire batch.
    - In this event, the captured user input can be used to control the script processing. Often the input will be stored in global variables for use in later events.
6. The **PreBefore** event fires.
    - This occurs for each item selected by the user.
    - Script logic can be added to this event to show information or gather input applicable to the item being processed.
7. The Confirmation Page for the item is shown.
8. The **Before** event fired.
    - This occurs for each item selected by the user.
    - In this event, the captured user input can be used to control the script processing for the item being processed.
9. The actual operation is executed.
10. The **After** event is fired.
    - Script logic can be added to this event to show information applicable to the item that was processed.
11. If configured, an Information Page can be shown for the item.
12. The **Terminate** event is fired.

- This occurs once for the entire batch.
- Script logic can be added to this event to show information applicable to the entire batch; however, you cannot make changes to the batch at this time.

13. If configured, an Information Page can be shown for the batch termination.
    - This occurs once for the entire batch.
    - Only summary information is shown. It is not possible to gather user input in this page.

Supported Events

The following VBScript events support confirmation pages.

Supported VBScript Events

Event Type	Event	Description
Custom	PreExecute	Used to add a confirmation page before a custom command has been executed.
Custom	PreInitialize	Used to add a confirmation page before a custom command is initialized.
Custom	Terminate	Occurs after the command has executed. This event does not support making changes to the batch, but you can use it to display information about the batch.
Document Copy / Move	PreBeforeCopyWithReferences	Occurs before the BeforeCopyWithReferences event. This event is used to add a confirmation page before the **Copy with References Wizard**.
Document Copy / Move	PreInitializeCopy	Used to add a confirmation page before the InitializeCopy event. This event can be used as part of a wizard.
Document Copy / Move	PreBeforeCopy	Used to add a confirmation page before the BeforeCopy event. This event can be used as part of a wizard. PreBeforeCopy is not called when executing Copy with References. Copy with References will call PreBeforeCopyWithReferences.

Event Type	Event	Description
Document Copy / Move	TerminateCopy	Occurs when a copy event is terminated. Applies only to **Copy** and **Paste** commands. This event does not support making changes to the batch, but you can use it to display information about the batch.
Document Generic	PreInitializeChangeDocumentType	Used to add a confirmation page before the InitializeChangeDocumentType event. This event can be used as part of a wizard.
Document Generic	PreBeforeChangeDocumentType	Used to add a confirmation page before the BeforeChangeDocumentType event. This event can be used as part of a wizard.
Document Project Copy	PreInitializeReleaseToMaster	Used to add a confirmation page before the InitializeReleaseToMaster event. This event can be used as part of a wizard.
Document Project Copy	PreBeforeReleaseToMaster	Used to add a confirmation page before the BeforeReleaseToMaster event. This event can be used as part of a wizard.
Document Workflow	PreBeforeRevokeWF	Occurs when a document type workflow is revoked by a user. This event is used to add a confirmation page before the revoke a document workflow action.
Document Workflow	PreInitializeRevokeWF	Occurs when a document type workflow is revoked by a user. This event is used to add a confirmation page before initializing the action.
Document Workflow	TerminateRevokeWF	Occurs when a document type workflow is revoked by a user. This event does not support making changes to the batch, but you can use it to display information about the batch.
Workflow Definition	PreInitializeExecuteTransition	Occurs at batch-level, before a workflow transition is initialized. This event can be used to add a confirmation page before initializing the transition.
Workflow Definition	InitializeExecuteTransition	Occurs at batch-level, when a workflow transition is initialized. This event applies the parameters specified in the confirmation page for the batch.

Event Type	Event	Description
Workflow Definition	PreBeforeExecuteTransition	Occurs at document-level, before a workflow transition is executed. This event can be used to add a confirmation page for an individual document, before initializing the transition. The document name is visible to the user.
Workflow Definition	BeforeExecuteTransition	Occurs at document-level, when a workflow transition is executed. This event applies the parameters specified in the confirmation page for the document.
Workflow Definition	AfterExecuteTransition	Occurs at document-level, after a workflow transition is executed. Any changes to the event parameters are ignored.
Workflow Definition	TerminateExecuteTransition	Occurs at document-level and ends the execution of the workflow transition. Any changes to the event parameters are ignored. This event does not support making changes to the batch, but you can use it to display information about the batch.

Relevant Properties and Methods

The following properties and methods are used with confirmation pages. The **Confirmation** method is used to trigger the confirmation page, while the rest of the properties and methods are used to configure the layout of the confirmation page.

The properties and methods used to format the confirmation page are displayed from top to bottom in the confirmation dialog according to the order in which they are called in the script.

Properties and methods usable in confirmation pages

Object	Type	Name	Purpose	Description
Batch	Property	ConfirmationTitle	Formatting	Allows you to set the title of the confirmation dialog box.
Batch	Method	Confirmation	Trigger	The method used to call your custom confirmation page.
Batch	Method	AskConfirmation	Formatting	Allows you to add **Yes** and **No** buttons to the confirmation page.

Object	Type	Name	Purpose	Description
Batch	Method	AskInput	Formatting	Allows you to add an editable text field to the confirmation page.
Batch	Method	ShowInfo	Formatting	Allows you to add static text to the confirmation page.

Procedures

You can add confirmation pages by using the VBScript Editor. For most confirmation pages, you will use the editor which is accessible from the toolbar at the top of the Configurator.

However, for custom commands, you also have the option to add confirmation pages to the script you created for the command. There is no functional difference between adding confirmation pages for custom commands to the VBScript editor or to the command script itself.

Add Confirmation Pages to VBScript

To add confirmation pages to a command or wizard:

1. In Configurator, click the **Edit Events** button  in the toolbar at the top of the screen.

   The VBScript Editor opens.
2. Enter your custom script.

   Learn more about creating and editing event procedures.

   The contents of the confirmation dialog will be ordered from top to bottom according to the order in which they are called in the script.
3. Click **OK**.

Add Confirmation Pages to Custom Command record

To add confirmation pages to an individual custom command record:

1. In Configurator, expand **Commands** in the configuration tree to display the existing commands.
2. Select the command you want to configure.
3. Click **Edit**.

   For other configuration options for custom commands, see Create And Edit Custom Commands.
4. Click the **Meridian Enterprise Script Editor** button  to build a VBScript expression.

   The VBScript editor opens.
5. Enter your script into the editor.

   Learn more about creating and editing event procedures.

   The contents of the confirmation dialog will be ordered from top to bottom according to the order in which they are called in the script.
6. Click **OK**.
7. Click **Apply**.

Meridian Enterprise Command Identifiers

The command identifiers (strings) that may be used as described in DocGenericEvent_ PrepareCommand event are listed below.

Identifiers

- Add to Collection
- Add to WorkList
- AttachPart
- Change Document Type
- Change Folder Type
- Change Manager
- Change ToDo Person
- Compare
- Create Draft Version
- Delete
- DeletePart
- DetachPart
- DiscardFromProject
- Draft Print
- LockMasterDocument
- MergedWithMaster
- New FileName
- NewRevision
- Off-line prepare
- Print Preview
- Properties
- ReassignManager
- ReassignProjectManager
- ReassignToDoPerson
- ReleaseToMaster

- Reroute
- RerouteProject
- Revoke document
- Revoke Draft Version
- Send Mail
- Set Field
- Set XRefs
- Show Versions
- Show XRefs
- Submit Draft Version
- SupersedByMaster
- Sync file cache (Local Workspace)
- Sync properties from database to file
- Sync properties from file to database
- Sync references from file to database
- Unlock file cache (Local Workspace)
- Unlock from briefcase
- Unlock from GCF
- UnlockFromProject
- UnlockMasterDocument
- Update thumbnails
- Update Title Block
- View

TagExtractor Component

Meridian Enterprise provides functionality for managing intelligent AutoCAD and MicroStation drawings that contain equipment and functional location tags. The standard functionality can be configured as described in the *Configure Asset Management* section of the *Meridian Enterprise Configuration Guide*. This functionality can be extended with VBScript programming.

The TagExtractor component is a dynamic link library (`TagExtractorUtl.dll`) from which you can create an automation object in scripting that exposes powerful properties and methods that you can use to implement your own tag functionality. The DLL is installed in the Meridian Program folder and can be used right away. For additional background on automation objects, see Automation Objects.

Using the TagExtractor component requires an Asset Management Client (M--AME) license for each user that will execute the scripting.

The following topics describe the various object and their properties and methods that are provided by the TagExtractor component.

TagExtractor Object

The **TagExtractor** object is created by the TaxExtractor component as described in TagExtractor Component.

You can create a **TagExtractor** object with the AMCreateObject Function as shown in the following example code.

```
Dim TagExtractor
Set TagExtractor = AmCreateObject("IcTagExtractor.IcTagExtractor")
If TagExtractor Is Nothing Then
  'Show error message and exit
Else
  'Do work here
End If
'Close and destroy when done
TagExtractor.Close
Set TagExtractor = Nothing
```

Unlike the base VBScript objects, the **TagExtractor** object properties, methods, and result codes do not appear in **Object Browser** of the Meridian Enterprise Script Editor.

TagExtractor Object Methods

The **TagExtractor** object provides the following methods.

CreateHotSpotList Method

Creates hotspot bounding box data for the tags found in the current drawing.

Syntax

```
CreateHotSpotList() As Integer
```

Return Value

One of the codes listed in TagExtractor result codes.

Remarks

Hotspot data is stored in the vault separate from the drawing file.

Example

```
Dim ResCode
ResCode = cTagExtractor.CreateHotSpotList
```

Open Method

Opens an AutoCAD or MicroStation drawing from which to extract the tags that it contains.

Syntax

```
Open(FileName As String) As Integer
```

Parameters

Name	Description
FileName	Fully-qualified path of the file in the local workspace to open.

Return Value

One of the codes listed in TagExtractor Result Codes.

Example

```
Dim ResCode
ResCode = TagExtractor.Open(sFileName)
```

SetAdditionalFilter Method

Sets a path filter to search for tag proxy objects in the vault.

Syntax

```
SetAdditionalFilter(iFilterType As Integer, sVaultPath As String)
```

Parameters

Name	Description
iFilterType	Specifies the filter type. The only supported type is parent folder. Always set this parameter to **1**.
sVaultPath	Vault path (recursive) in which to limit tag searches.

Return Value

This method returns no values.

Remarks

By default, the SyncTagReferences method searches for tags in the entire vault using the settings of the **Tag document type** and **Tag indication property** options (on the **AMM Settings** page in the vault configuration) and the tag number. Use this method to search for tags only in the specified vault folder (and its sub-folders) without using the **Tag document type** setting.

Examples

To set a filter:

```
TagExtractor.SetAdditionalFilter 1, sVaultPath
```

To disable a filter:

```
TagExtractor.SetAdditionalFilter 1, Nothing
```

SyncTagReferences Method

Searches for tag proxy objects in the vault and creates Meridian references between them and the current drawing.

Syntax

```
SyncTagReferences() As Integer
```

Return Value

One of the codes listed in TagExtractor Result Codes.

Remarks

The direction of the references is as set in the asset management settings of the vault. The scope of the tag search can be limited by the SetAdditionalFilter method.

Example

```
Dim ResCode
ResCode = TagExtractor.SyncTagReferences
```

SaveLogToFile Method

Creates a log file that contains the results of the SyncTagReferences method.

Syntax

```
SaveLogToFile (sLogFile As String) As Integer
```

Parameters

Name	Description
sLogFile	Fully-qualified path where to save the log.

Return Value

One of the codes listed in TagExtractor result codes.

Remarks

Call this method after performing the tag extraction with the SyncTagReferences method.

Example

```
Dim ResCode
ResCode = TagExtractor.SaveLogToFile(sLogFile)
```

TagCollection Method

Returns a collection of all tags found in the current drawing.

Syntax

```
TagCollection(oResCode As Object) As TagCollection
```

Parameters

Name	Description
oResCode	Result code object in which to return the result of the method.

Return Value

A collection of **Tag** objects.

Remarks

To get a collection of all tags with a specific tag type, use the **TagExtractor**.TagCollectionEx method.

Example

```
Dim TagCollection
Dim ResCode
Dim Tag

Set TagCollection = TagExtractor.TagCollection(ResCode)
'Iterate through the collection
For Each Tag In TagCollection
  'Do work here
Next
```

TagCollectionEx Method

Returns a collection of all tags of the specified tag type found in the current drawing.

Syntax

```
TagCollectionEx(oResCode As Object, iCollectionType As Integer) As
TagCollection
```

Parameters

Name	Description
oResCode	Result code object in which to return the result of the method.
iCollectionType	Integer that represents the type of tags to find: **1**: All tags **2**: Resolved tags only **3**: Unresolved tags only

Return Value

A collection of **Tag** objects.

Remarks

To get a collection of all tags regardless of the tag type, use the **TagExtractor**.TagCollection method.

Example

```
Dim TagCollection
Dim ResCode
Dim Tag

Set TagCollection = TagExtractor.TagCollectionEx(ResCode,
iCollectionType)
'Iterate through the collection
For Each Tag In TagCollection
  'Do work here
Next
```

TagCollection Object

A **TagCollection** object is returned from by the **TagExtractor**.TagCollection method, **TagExtractor**.TagCollectionEx method, or represented by a **TagExtractor**.TagIterator object. It contains a collection of Tag objects.

TagCollection Object Properties

The following sections describe the **TagCollection** object properties.

TagPrefix Property

Gets the value of the property specified for the **TagPrefixPropertyDef** setting in the vault.

Syntax

```
TagPrefix As String
```

Remarks

For information on configuring tag prefixes and postfixes, see *Configure Tag Name Prefixes And Suffixes* in the *Meridian Enterprise Configuration Guide*.

TagPostfix Property

Gets the value of the property specified for the **TagPostfixPropertyDef** setting in the vault.

Syntax

```
TagPostfix As String
```

Remarks

For information on configuring tag prefixes and postfixes, see *Configure Tag Name Prefixes And Suffixes* in the *Meridian Enterprise Configuration Guide*.

Tag Object

The **Tag** object is a member of a collection returned by the **TagExtractor**.TagCollection method, **TagExtractor**.TagCollectionEx method, or represented by a **TagExtractor.TagIterator** object described in TagIterator Object.

Tag Object Properties

The **Tag** object provides the following properties.

Tag Property

The value of the drawing tag.

Syntax

```
Tag As String
```

Example

```
Dim TagValue
TagValue = Tag.Tag
```

TagType Property

The type of the drawing tag.

Syntax

```
TagType As String
```

Example

```
Dim TagType
TagType = Tag.TagType
```

BlockList Property

A collection of the blocks in the current drawing that contains the tag represented by the current **Tag** object.

Syntax

```
BlockList As Object
```

Remarks

Each block in the collection has only one (string) property: **BlockName**.

Example

```
Dim Blocks
Blocks = Tag.BlockList
```

ObjectList Property

A collection of the tag proxy objects in the current vault that represent the current **Tag** object.

Syntax

```
ObjectList As Object
```

Remarks

This collection is typically empty or contains only one tag proxy object. Each tag proxy object has only one (string) property: **ObjectID** that corresponds to the Meridian document **ID** property.

Example

```
Dim TagObjects
TagObjects = Tag.ObjectList
```

TagIterator Object

The **TagIterator** object provides easier access to a set of **Tag** objects than a collection returned by the TagCollection method and TagCollectionEx method. It provides a forward-only way to step through a collection and immediately retrieve **Tag** properties without instantiating a **Tag** object.

You can create a **TagIterator** object as shown in the following example code where `iCollectionType` is an integer that represents the type of tags to find:

- **1**: All tags
- **2**: Resolved tags only
- **3**: Unresolved tags only

```
Dim TagIterator
Set TagExtractor = TagExtractor.TagIterator(iCollectionType)
```

TagIterator Object Properties

The following sections describe the **TagIterator** object properties.

IsValid Property

True if the current **TagIterator** collection is not empty. **False** if it is empty.

Syntax

```
IsValid As Boolean
```

Count Property

Gets the number of tags in the **TagIterator** collection.

Syntax

```
Count As Long
```

Tag Property

Gets the (string) value of the current **Tag** object in the **TagIterator** collection.

Syntax

```
Tag As String
```

Remarks

This property is equivalent to the Tag property of the **Tag** object.

TagType Property

Gets the type of the current **Tag** object in the **TagIterator** collection.

Syntax

```
TagType As String
```

Remarks

This property is equivalent to the TagType property of the **Tag** object.

BlockArray Property

Gets an array of the block names in the current drawing that contains the tag represented by the current **Tag** object in the **TagIterator** collection.

Syntax

```
BlockArray As String
```

Remarks

This property is equivalent to the BlockList property of the **Tag** object.

ObjectArray Property

Gets a collection of the tag proxy objects in the current vault that represent the current **Tag** object in the **TagIterator** collection.

Syntax

```
ObjectArray As Long
```

Remarks

This property is equivalent to the ObjectList property of the **Tag** object.

TagIterator Object Methods

The following sections describe the **TagIterator** object methods.

Start Method

Sets the index position to the first **Tag** object in the current **TagIterator** collection.

Syntax

```
Start
```

Return Value

This method returns no values.

Remarks

Use the **TagIterator**.Step method to advance the index position to the next available **Tag** in the current **TagIterator** collection.

Example

```
Dim TagExtractor
Set TagExtractor = AmCreateObject("IcTagExtractor.IcTagExtractor")
Dim TagIterator
Set TagIterator = TagExtractor.TagIterator(iCollectionType)

If TagIterator.IsValid Then
  'Iterate through the collection
  TagIterator.Start
  Do
    'Do work here
  Loop While TagIterator.Step
End If
```

Step Method

Advances the index position to the next available **Tag** in the current **TagIterator** collection.

Syntax

```
Step As Boolean
```

Return Value

Returns True if the next **Tag** exists.

Remarks

Use the **TagIterator**.Start method to set the index position to the first available **Tag** in the current **TagIterator** collection.

Example

```
Dim TagExtractor
Set TagExtractor = AmCreateObject("IcTagExtractor.IcTagExtractor")
Dim TagIterator
Set TagIterator = TagExtractor.TagIterator(iCollectionType)

If TagIterator.IsValid Then
  'Iterate through the collection
  TagIterator.Start
  Do
    'Do work here
  Loop While TagIterator.Step
End If
```

TagExtractor Result Codes

This **TagExtractor** methods return the following result codes that you can use in your scripting logic to control procedure flow.

TagExtractor result codes

Code	Description
-1	Unspecified error
0	Operation succeeded
1	One or more libraries needed to access the document data were not found
2	Specified file was not opened
3	Specified file cannot be opened
4	Documents of this version are not supported
5	Document that was opened is not from the vault
6	Some objects are not found. A list of unresolved tags may be saved to file by the SaveLogToFile method.
7	Cannot save log to file
8	Empty log
9	Asset settings are not found
10	No license found
11	License is invalid
12	Invalid patch area
13	Patch area is expired

Publishing And Rendering Options

When documents are registered by the source vault's VBScript event handlers, publishing options and rendering options can be specified that modify the publishing job. These options can be set using the **RegisterDocument** method described in RegisterDocument method.

The Meridian Enterprise system link supports the options described in the following table when the destination system is another Meridian Enterprise vault. The Microsoft SharePoint system link also supports publishing options, however they must be specified in a customized SharePoint workflow template. The publishing options are not supported by the Windows file system link.

> **Note:**
> If an option accepts a value and the value is empty, the publishing job settings are used. If the value is invalid (missing < or > characters), the job will fail.

Publishing options

Option	Description
CDWF *<WorkflowDefinitionName ;WorkflowStateName>*_	The document is published to the destination system immediately into the specified workflow state.
DATAONLY	The document content is not transmitted, only the property values. This option is typically used together with the **_NORENDER_** option.
DELETE OR _DELETE_*<FileName>*	The document is deleted from SharePoint libraries when the Meridian Enterprise source document is deleted. If the destination filename is not specified, the name of the document registered for publishing is used. Due to the risk of accidental data loss, this option is only available if the **EnableRemoveSharePointTarget** option is set to `true` in the following file. Be certain that you want to use this option before enabling it. `C:\ProgramData\BlueCieloECM\EnterpriseServices\PublishingCapability.dat`
LIB_*<Title>*	The title of the SharePoint library to publish to. If specified, the value replaces the **Library** option that is configured in the publishing job. **Note:** This option is supported by the SharePoint publishing module only.

Option	Description
NOREDLINES	Prevents the publishing of redlines.
NORELEASE	The document is not released in the destination vault after publishing.
OVERWRITE	If any revisions of the document exist in the destination system, they are deleted before the current revision from the source system is published. **Note:** This option is supported by the SharePoint publishing module only.

Option	Description

PUBLISHONLY *<PropertySet.PropertyName;PropertyValue>*_	A safeguard for certain configurations in which a document can be rendered and published before changes to the document are saved in the vault. This can result in an invalid rendition in the destination system.
	Specify the name of a custom property for the source system link to check for a specified value before publishing the document. If the property does not contain the specified value, the document is not published by that attempt, the text **Document is not ready for publishing** is saved in the publishing log, and the property is rechecked on each retry.
	Scripting is responsible for setting the value prior to publishing and resetting the value after publishing unless the value is unique for each document.
	`Sub DocWorkflowEvent_AfterChangeWFState(Batch, _`
	`  SourceState, TargetState, Person, Comment)`
	`  Dim PublishNo`
	
	`  If (SourceState = AS_WF_UNDERCHANGE) And _`
	`  (TargetState = AS_WF_RELEASED) Then`
	
	`  'Set Property for _PublishOnly_ option`
	`  Document.PublishNo = CreateGUID`
	`  PublishNo = Document.PublishNo`
	
	`  ' Send document to Publisher`
	`  Set Publisher = New PublisherScriptObject`
	`  'Register current document for publishing`
	`  Call Publisher.Queue.RegisterDocument("", _`
	`  "398328B7", Document.ID, _`
	`  , "_PUBLISHONLY_<Custom.PublishNo;" & _`
	`  PublishNo & ">_" ,_`
	`  , Vault.User.Name, , _`
	`  "Custom.PublishStatus")`
	`  'Release the script object.`
	`  Set Publisher = Nothing`
	`  End If`

Option	Description
	``` End Sub ``` We recommend setting the value to the document revision number using `Document.VersionID`, a vault sequence number using `Vault.Sequence(`*`Name`*`)`, or a globally unique ID (GUID). Following is an example of getting a GUID in VBScript: ``` Function CreateGUID Dim TypeLib Set TypeLib = CreateObject("Scriptlet.TypeLib") CreateGUID = Left(TypeLib.Guid, 38) End Function ```
_RENDERLAYOUTS__ *<LayoutName1;LayoutName2;...LayoutNameN>*_	Renders each of the specified layouts to a separate rendition. This option is supported for AutoCAD and Revit drawings rendered with the AutoCAD rendering module, and Teigha DWG rendering module only. Specify the layout names to render in a semicolon-delimited list. By default, the name of the rendition is a combination of the source file name and the layout name (*<FileName_LayoutName>*). You can specify a different name with a VBScript expression as the value of the **Use property value from source document** option of the job configuration. Use the VBScript property **Document.LayoutsNames** to retrieve the layout names for the current document. The title block attribute or sheet property names must be configured in the appropriate application link settings of the vault configuration as described in the *Application Integration* section of the *Meridian Enterprise Configuration Guide* . If the source document and the renditions reside in the same Meridian Enterprise vault, the rendition type **Layout Rendition Reference** (**RTRenditionReference**) will be created in the vault if it does not already exist. That rendition type will be used to create references between the source document and the renditions. The name of each reference is the layout name.

Option	Description
RESERVE OR _RESERVE_*<FileName>*	Checks the document out in the destination system, creating a new major or minor revision depending on the system configuration. The revision can be revoked and the new revision deleted by the **_UNRESERVE_** option. **Note:** • This option is supported by the SharePoint publishing module only. • If the destination filename is not specified, the name of the document registered for publishing is used. • The document will be checked out by the Meridian Enterprise system account. • Do not use with the **_UNRESERVE_** or **_DELETE_** options in the same operation or only the first option will be applied. • The **Comment** (workflow log) property is not updated by this option.
RETIRE	The document is not published to the destination system, but the existing document in the destination vault is retired.
RETRY_*<Count>*	If the job fails for any reason, restart the job in the queue **<Count>** times or until the job succeeds, whichever comes first. This option overrides the **Number of retries if publishing fails** setting described in *Configuring the Publishing Options* in the *Meridian Enterprise Server Administrator's Guide*.
SITE_*<URL>*	The URL of the SharePoint site and subsite (if applicable) to publish to. If specified, the value replaces both the **Site address** and the **Subsite** options that are configured in the publishing job. **Note:** This option is supported by the SharePoint publishing module only.
UNRESERVE OR _UNRESERVE_*<FileName>*	Revokes the checkout done by the **_RESERVE_** option of the specified document in the destination system and deletes the unreleased revision. **Note:** • This option is supported by the SharePoint publishing module only. • If the destination filename is not specified, the name of the document registered for publishing is used. • Do not use with the **_UNRESERVE_** option in the same operation or only the first option will be applied.

The rendering modules support the options described in the following table.

Rendering options

Option	Description
NOPOSTRENDER	Disables all rendition post-processing actions (for example, watermarks or signature pages).
NORENDER	Disables all document rendering and an empty file is published to the destination vault. This option is typically used together with the **_DATA_ONLY_** option.
PRINTING QUALITY	Allows you to get and set the print quality to use with PDFTron. You can set a value between 1 and 15. Higher values result in higher quality, but takes longer to render. Values between 3 and 6 are suitable for most purposes. Beyond 6, quality increases by diminishing amounts. If no value is set, the default value of 1 is used. The property exists within the `BCRenditionPropertySet`. For example, `Document.Property("BCRenditionPropertySet._PRINTING_QUALITY")`.
REVITSETUP *<ExportSetupName>*_	Uses the parameters configured in the specified export setup to render the Revit project instead of the options configured in the publishing job. If the export setup cannot be found, then the configured options will be used.
ExternalUpdateProperties	If you use the Teigha rendering engine, this setting allows Meridian to recover from attempting to render a corrupt or invalid document. If a third-party rendering component fails, we remove the document from the rendering queue. A message appears in the process log that informs the user that the process has failed, and then Meridian proceeds to the next document in the queue. To enable this setting: 1. Navigate to the following file: `C:\ProgramData\BlueCieloECM\EnterpriseServices\PublishingCapability.dat` 2. Change the value of the **ExternalUpdateProperties** setting to **true**. 3. Save the file.

Glossary

A

Active Directory

A Microsoft directory service that provides central authentication and authorization services for Windows-based computers.

AMFS

The InnoCielo File System service that makes vaults available through the Windows file system.

approved

A workflow status that indicates that a document that has been approved for reproduction, distribution, manufacture, or construction.

archive

When used as a noun, a repository of obsolete documents kept for possible future reference. When used as a verb, the process of exporting obsolete documents from a repository.

assign to work area

The process of creating a copy of a document in a work area to isolate its changes from the original revision in the Main area.

attribute

When used to describe a file system, it is a property of a file such as Hidden, System, or Read Only. When used to describe an AutoCAD drawing, a named object in a drawing that is included in a block definition and used to store alphanumeric data.

audit log

A system-generated record of the date and time of user actions that create, modify, or delete critical business data.

audit trail

A system-generated record of the date and time of user actions that create, modify, or delete critical business data.

authorization key

The ten character hexadecimal code generated by BlueCielo ECM Solutions that authorizes a software license indefinintely. Authorization keys are generated based on

the license serial number, license key, and return key specific to each installation.

B

baseline

When used to describe Meridian Enterprise, a named moment in time in the history of a vault, such as a milestone.

Basic Authentication

A method designed to allow a web browser, or other client program, to provide credentials – in the form of a user name and password – when making a request from a server.

briefcase

An Accruent portable document package. A briefcase is a single file in an archive format that may contain multiple discrete documents. Briefcases may be in open standard formats such as ZIP and RAR, the Accruent BRC format, or custom formats. A briefcase may also contain document metadata in a data file and, in the Accruent BRC format, document redlines.

C

client

A computer, object, or program that obtains data or services from a server.

COM

Component Object Model - an interface standard for software componentry by Microsoft used to enable interprocess communication and dynamic object creation by programs.

content

The electronic data associated with a document.

content indexing

The process of extracting and indexing text data from documents for full-text searching. See also "full text search".

context

When used to describe Meridian Enterprise, a frame of reference comprised of a specific compartment of a vault and a moment in time for viewing the content of a vault.

criterion

A search filter condition comprised of a property name, operator, and value.

current

In general, the object that a user has selected or an object that is currently within the scope of a programming expression. When used to describe the history of a document, the latest revision of a document, which might not yet be released.

D

data source

An external data file or database that provides data that is presented by or imported into a Accruent system.

database

A structured set of document metadata used by a Accruent system. The database may be managed by Hypertrieve, SQL Server, or Oracle depending on the system.

DB

An Oracle database.

DCOM

Distributed Component Object Model - a Microsoft proprietary technology for software components distributed across several networked computers to communicate with each other.

derive

To create a new document based on an existing document. Also the name of a Meridian Enterprise command.

destination state

The state of a Meridian Enterprise workflow that follows a workflow transition.

details page

A type of property page that displays the properties of a document.

digital signature

A digitized image of a person's handwritten signature. See also "electronic signature".

discard

When used to describe Meridian Enterprise, to cancel the process of revising a document and delete the file copy that is being edited.

document

Information recorded on a medium (paper, digital media, and so on) for communication to others.

document controller
A person within a facility owner/operator organization that is responsible for the management of project documents.

document type
A classification of documents that share one or more document management characteristics such as format, purpose, or security.

document type workflow
A predetermined sequence of steps through which a document must be processed to generate a new approved revision in Meridian Enterprise. The workflow is defined by the document type from which the document was created.

document view
A view of a Meridian Explorer repository that displays documents.

document workflow interlock
A rule consisting of specific document types and property filters that limit when a project's workflow may proceed. Interlocks are configured by a System Administrator with the Meridian Enterprise Configurator application. Interlocks are available only with the Meridian Advanced Project Workflow Module.

domain controller
A server that responds to security authentication requests (logging in, checking permissions, and so on) within a Windows Server domain.

dynamic collection
A Meridian Enterprise saved search in which the search criteria are reevaluated and the results are updated whenever the collection is displayed.

E

e-signature
An electronic indication that a person adopts the contents of an electronic message. See also "digital signature".

ECM
Engineering Content Management. Content management as it applies specifically to engineering.

EDM
Engineering Document Management. Document management as it applies specifically to engineering documents.

effectivity
An attribute of a Meridian Enterprise property that determines when changes to its value apply within the life cycle of a document.

electronic signature
An electronic indication that a person adopts the contents of an electronic message. See also "digital signature".

environment
An organization's overall computing platform.

Explorer view
The view of a Meridian vault that displays documents organized within the Field-Path Relation.

external page
A type of property page that displays a web page that is hosted on a different information system.

F

Field-Path Relation
A hierarchical structure defined by properties that determines the folder structure of a vault and the locations of documents according to the values of the properties.

Folders view
The view of a Meridian Enterprise vault that displays documents organized by the vault's Field-Path definition.

FS
An acronym for file system.

full-text search
A method of searching for text contained in document content as opposed to searching document metadata. See also "content indexing."

G

grid view
Name of a Meridian Explorer view mode that displays search results in tabular format.

GUID
An acronym for Globally Unique Identifier.

H

history

A configurable option of a Meridian vault that causes it to save changes to documents and metadata over time. Allows users to view prior revisions of documents and their property values at specific moments in the past.

History mode

A configurable option of a Meridian vault that causes it to save changes to documents and metadata over time. Allows users to view prior revisions of documents and their property values at specific moments in the past.

HTTP

An acronym for Hypertext Transfer Protocol.

hybrid drawing

A drawing composed of both vector graphics and raster image files.

I

import

The process of creating a new file in a vault from a file outside the vault or repository .

inactive user

A user account that has been deactivated. The account is not deleted but the user cannot use the application. The account can be reactivated later.

initiate

The act of starting a new revision of a document by performing the first step of a workflow.

issue code

The name of a Meridian Transmittal Management Module property that stores a standard keyword or phrase. The issue code describes the reason why a transmittal was issued.

L

layout

A configurable arrangement of items on a form or page.

LDAP

An acronym for Lightweight Directory Access Protocol.

Local Workspace

A portion of disk space on a user's computer reserved for caching documents when they are viewed or edited. Using Local Workspace improves performance when working with very large documents. Meridian Enterprise's local workspaces can be synchronized with the vault at a configurable interval.

lookup list

A list of predetermined values for a property that is presented to the user for selection. Lookup lists can be either managed in the application or linked to an external database or query.

M

Main area

The area of a Meridian vault where released documents reside.

manager

The Meridian user who initiated the current document workflow and the only person with permission to change the current To-Do Person.

master document

In Meridian Enterprise, a released document from which one or more project copies are made that become either independent documents or new revisions of the master document. Master documents are designated by Meridian Enterprise and the Meridian Advanced Project Workflow Module. In Accruent Project Portal, a document to which a master file is attached.

Meridian Enterprise Server application server

The Meridian Enterprise Server data access and business services running on a server computer. May also be used to refer to the server hosting the services.

Meridian Explorer client

The Meridian Explorer application installed on the Meridian Enterprise Server web server.

metadata

Information that classifies, supplements, or describes a document. Metadata is commonly used to find documents as opposed to searching for data within documents (see “full-text search”). Metadata may also be used for a variety of other purposes.

N

NAT

An acronym for Network Address Translation.

Navigation view

A view of a Meridian vault that displays documents organized in a hierarchical structure according to a predefined set of properties.

O

OS

An acronym for operating system.

OU

An acronym for organizational unit.

P

package

A set of files that are used together for a common purpose. The files are often bound together in a single archive file for convenience when transporting, such as .zip and .msi files. Examples of file packages are software distribution packages and submittal packages. See also “briefcase.”

pane

A separate area of a split or single window used to display related data.

performance counter

Stores the count of specific program activities on a computer to conduct low-level performance analysis or tuning.

PowerUser

The Meridian desktop client software. Not related to the Windows administrative group Power Users.

PowerWeb

The Meridian Enterprise web browser-based client application.

preselection

A property filter that can be applied to a Meridian Explorer view to limit the number of visible items.

privilege

The right of a user to view specific data or execute a specific command. Privileges are assigned by a System Administrator.

project copy

A copy of a master document made for the purpose of creating a new independent document or creating a new revision of the master document. Project copies can be created in Meridian Enterprise.

project definition

A template used to create special folders in a vault that can represent design project processes. Project definitions are configured by a System Administrator with the Meridian Enterprise Configurator application. Meridian Advanced Project Workflow Module project definitions consist of a custom folder type, a workflow, and optional project workflow interlocks or document workflow interlocks. A project definition may restrict folders from being created at the root of a vault and may restrict creation of subprojects (Meridian Advanced Project Workflow Module only).

project folder

A folder created from a project definition template.

project workflow

The workflow of a project folder as defined by the project definition template from which it was created. Configured by a System Administrator with the Meridian Enterprise Configurator application.

project workflow interlock

A rule comprised of specific sub-project folder types and property filters that is applied to a project or subprojects that limits when a project's workflow may proceed. Interlocks are configured by a System Administrator with the Meridian Enterprise Configurator application. Interlocks are available only with the Meridian Advanced Project Workflow Module.

property

Descriptive data used to identify, classify, and find documents. Properties are organized into related groups called property sets.

property navigation

A dynamic search method in which a user progressively reduces the number of documents found by selecting from additional property values.

property page
A secondary window, usually displayed with a tab, that displays the properties of an object such as a document.

property set
A group of related properties.

publish
To create a copy of a document in another information system, optionally in a different electronic format.

purge
To completely and permanently delete data from a system.

Q

query
A search command comprised of one or more search criteria often expressed in Structured Query Language (SQL) syntax.

Quick Change
A very simple document workflow consisting of only two steps, Start Quick Change and Release Quick Change that may or may not increment the document's revision number depending on the configuration of the document type.

R

reassign
To immediately assign a document to the current work area for additional changes after discarding or releasing the current revision.

recovery log
The log of vault documents that can be executed in order to export the documents from a vault to a specified location on the file system. The recovery log is created for use in the event of a critical disaster to provide continued access to documents.

redline
Corrections to a drawing made graphically on a copy of the drawing. Redlines can be created for electronic drawings with the InnoCielo viewer by a user with the appropriate security privileges.

reference
A link that represents a relationship between two documents. References can be created automatically by Meridian (for example, AutoCAD External Reference) or manually by a user.

reference type
A classification of references that share one or more document management characteristics such as purpose, source or destination document types, or security.

references page
A type of property page that displays the references of a document.

related documents page
A type of property page that displays the documents that are related to the selected object.

related tags page
A type of property page that displays the asset tags that are related to the selected document.

release
The final step (transition) of a Meridian Enterprise workflow. When describing project workflow, refers to a new revision of a master document that was created from the content of a project copy. When describing document type or workflow definition workflows, refers to a new revision of the document that was created by completing the document's workflow.

render
Rendition (noun) refers to a copy of a document in a format other than the original. Render (verb) refers to the process of creating a rendition.

rendition
Rendition (noun) refers to a copy of a document in a format other than the original. Render (verb) refers to the process of creating a rendition.

repository
The largest logical container of a document management system for storing documents and metadata. A repository commonly contains all of the documents for a single organization, division, department, workgroup, or other purpose, organized into folders and sub-folders. The fundamental container of a Meridian Explorer system.

result grid
A configurable grid view used to display documents or tags found by a search.

retire
To classify a document as obsolete and prevent it from being revised.

return code
A standard keyword or phrase that represents the reason why a submittal was issued.

review
The process of evaluating the accuracy and completeness of revisions to a document.

revision
A milestone in a document's history that represents approved information at particular point in time identified by a number or letter.

revisions page
A type of property page that displays a list of the revisions of a document.

revoke
The act of canceling revision of a working copy of a document and deleting the copy being edited.

role
A named set of privileges to which users or groups are assigned by an administrator.

RPC
Acronym for Remote Procedure Call.

S

saved search
A user-defined set of search criteria that is saved for future reuse.

scope
A Meridian Enterprise feature that limits vault functionality and the visible information to named sets. A scope can be selected by users to make the system easier to use or to gain access to different documents.

search layout
A configurable combination of repository navigation and search filter parameters used by Meridian Explorer.

server
A centralized computer or application that provides services to one or more client computers or applications in a network.

shared workspace
A special folder in a Meridian Enterprise vault that is used to store files to support multi-user applications. The vault folder is mapped to a shared network location outside the vault that is used instead of local workspaces on the users' computers. Meridian Enterprise synchronizes the contents of the shared network location with the vault folder. Configurable options control other behaviors specific to using a shared workspace.

shortcut bar
The name of the accordion control containing shortcuts to views, vaults, and baselines that can be displayed in the left pane of the Meridian Enterprise desktop application.

SID
An acronym for System Identifier. A name that identifies a specific instance of a running Oracle database.

SMTP
An acronym for Simple Mail Transport Protocol.

snapshot
A read-only copy of metadata made so that slower data backup processes can occur while the application continues writing to its data. Backing up a snapshot minimizes maintenance downtime.

source state
The state of a workflow that precedes a workflow transition.

SSL
An acronym for Secure Sockets Layer or Transport Security Layer.

SSL/TLS
An acronym for Secure Sockets Layer or Transport Security Layer.

static collection
Saved search results that are displayed without reevaluating the search criteria.

sub-project
A Meridian Enterprise project folder contained within another project folder that can represent a subordinate process. Subprojects are available only with the Meridian Advanced Project Workflow Module.

submit
When used to describe a document, means to check in the working copy of a document that is under revision. Equivalent to releasing a document from a workflow.

submittal

A package of documents received by an organization for review, reference, modification, or final delivery.

T

tag

A vault or repository record that represents a logical asset stored in a separate maintenance management system. The logical asset represents a physical asset that is present at a facility that is managed with the maintenance management system. A tag may reference one or more documents, or the reverse.

tag type

The document type that is configured for use as asset tags.

thumbnail

A small preview image that is shown to assist the user in identifying a file.

TLS

An acronym for Secure Sockets Layer or Transport Security Layer.

To-Do List

The name of a navigation view in Meridian Enterprise.

transaction isolation

A property in a database system that defines how and when the changes made by one operation become visible to other concurrent operations.

transition conditions

Property value filters and logical expressions that are evaluated to determine the validity of a workflow transition to be executed by a user.

transition equivalence

The equality of a Meridian Enterprise transition in one document workflow to a transition in another document workflow. Transition equivalence makes it possible to execute a transition for one document in a batch of documents and have it also execute transitions in the other documents within the batch even if the transitions don’t have the same name, source state, or destination state. Configured by a System Administrator with the Meridian Enterprise Configurator application. Transition equivalence is available only with the Meridian Advanced Project Workflow Module.

transmittal sheet
A cover letter for a submittal that lists the names and other property values of the documents that are included in the submittal. It might also include comments about the status of the documents or the project, instructions to the recipient, and a date by which a response to the submittal is due back to the sender.

U

unretire
To reverse the effects of retiring a document so that it can be revised.

URL
An acronym for Uniform Resource Locator used to specify Internet and intranet addresses.

V

vault
A Meridian repository for storing documents related by organization, division, department, workgroup, or purpose.

VBScript
The Visual Basic scripting language (Visual Basic Scripting Edition).

version
A document derived or copied from another document of the same revision.

VPN
An acronym for Virtual Private Network.

W

WAN
An acronym for wide area network.

watermark
Textual or graphic information overlaid on a printed document used to indicate authenticity or validity.

Web Access
The Meridian Enterprise web browser-based client application.

web client
A client application that is presented in a web browser.

Work Isolation mode
The vault setting that defines how and when the changes made by one user become visible to other concurrent users.

workflow
A predetermined sequence of steps used to produce standardized results.

working copy
A temporary copy of a document made for making changes as an alternative to document workflow.

workstation
A personal computer used by an individual in a network. A workstation is the client in a client/server system.

X

X-Ref
An AutoCAD drawing that is linked to, but not inserted into, the current drawing. Changes made to referenced drawings (X-Refs) are automatically displayed in the current drawing when the current drawing is opened.

Index

A

accounts
- user accounts See *user accounts*

ActiveX
- DocProjectCopyEvent
  - ValidateTargetFolder Event not supported 367

AMM See *Asset Management Module*

AMUIExtension.GetParentFolder 133, 233

API
- Meridian API constants 407

Application Integration
- debug 65

application links See *Application Integration*

arguments
- object arguments 415
- Prompt argument and RTF codes 418

Asset Management Module
- create hotspot bounding box data 431
- events 300
- Tag Object 438
- TagCollection Object 437
- TagExtractor 430
- TagExtractor Object 431
- TagExtractor Result Codes 445
- TagIterator Object 440

automatically generated messages 411

automation objects 413
- create 413
- TagExtractor component 430

B

batch
- events 295

breakpoints 64

C

CAD Link Events 311
- effects of custom scripting 62

command identifiers 428

configuration
- expressions (VBScript) 67
  - Meridian Enterprise Script Editor 57

confirmation pages 421
- Batch Object
  - AskConfirmation method 77
  - AskInput method 78
  - Confirmation method 78
  - ConfirmationTitle property 74
  - Input method 80
  - ShowInfo method 81
- Custom Command Events
  - PreExecute Event 320
  - PreInitialize Event 320
- Document Copy/Move Events
  - PreBeforeCopy 327
  - PreBeforeCopyWithReferences 327
  - PreInitializeCopy 328

PreTerminateCopy 332
Document Project Copy Events
DocProjectCopyEvent_ PreBeforeReleaseToMaster 364
DocProjectCopyEvent_ PreInitializeReleaseToMaster 363
Document Type Workflow Events
DocWorkflowEvent_*RevokeWF events 371
Workflow definition events
ExecuteTransition events 404
connection string
Vault.ExecSQL method 232
constants 250
AS_BRC_IMPORT_ACTION 407
AS_CALLREMOTE_FLAGSS 407
AS_CE_RULE 407
AS_CI 407
AS_CLIENTID 407
AS_CMD_MODE 317, 407
AS_CMD_STATE 407
AS_CMD_STATUS 318
AS_CONFIRM_ACTION 407
AS_FEATURES 407
AS_GROUP_COLUMNS 407
AS_HOTSPOTS_TYPE 408
AS_HYBRID_ACTION 352-353, 408
AS_IMPORTDETAILS 408
AS_IMPORTTYPE 345, 377, 408
AS_INCLUDE_CONTENT_OPTIONS 132, 408
AS_LOG_FLAGS 91
AS_LOGFLAGS 408
AS_MAPIMSG_RECIP_TYPE 174, 408
AS_MAPIMSG_SEND_FLAGS 170, 186, 408
AS_MOVE_OPTIONS 127, 408
AS_MsgBoxResult 408
AS_MsgBoxStyle 291, 408
AS_NEWFOLDER_OPTIONS 160, 163, 408
AS_PCLOCK 408
AS_PORTAL_OPTIONS 132, 408
AS_PRIVILEGES 408
AS_PROJITEM_FLAGS 408
AS_PROP_CMD 346, 408
AS_PWF_STATUS 408
AS_REFRESHFLAG 409
AS_SUBMITTAL_STATUS_FLAGS 409
AS_SUBMITTAL_STATUS_VALUES 409
AS_TQTYPE 409
AS_TRANSMITTAL_STATUS_FLAGS 409
AS_TRANSMITTAL_STATUS_VALUES 409
AS_URL_FLAGS 230, 409
AS_USER_COLUMNS 409
AS_WA_STATEFLAG 409
AS_WATERMARK_COLORS 409
AS_WATERMARK_STYLES 409
AS_WATERMARK_TITLE_STYLES 409
AS_WF_STATE 112, 370, 409, 411
AS_WFINTERLOCK_LOCATION 409
AS_WORKFLOW_STATE_TYPE 409
AS_WORKFLOW_TRANS_RES 120, 409
IC_OPERATOR 409
IC_SHOWWINDOW 409

creating
event procedures 298
custom
client extensions 62
commands
effects of custom scripting 62
events 317
custom events
Custom Page Events 321

D

.dat files
PublishingCapability.dat
VBScript 446
Database Import Wizard
debug 65
date properties 411
debugging
enable VBScript logging 65
host applications 65
VBScript in PowerWeb 63
VBScript in Visual Studio 63
destination vaults 395
DLL files
AmEdmUI.dll 86
TagExtractorUtil.dll 430
document generic events 334
DocGenericEvent_BeforeNewDocument event 410
Document Import Tool
debug 65
Document shortcut menu 431
documents
clearing property values of copied documents 410

E

editing
event procedures 298
email
automatically generated messages 411
event handler
about 293
debugging 63
errors
Access is denied if not customized 341
event procedures
command identifiers 428
creating and editing 298
events
about
event procedures 293, 298
limiting generated events 299
names of 294
order of 296
effects of custom scripting 62
types
Asset Management Events 300
Batch Events 295
CAD Link Events 311
Custom Command Events 317
Custom Page Events 321
Document Copy/Move Events 323

Document Generic Events 334
Document Hybrid Events 352
Document Project Copy Events 355
Document Type Workflow Events 368
Document Working Copy Events 373
Folder Generic Events 376
Package Events 379
Project Workflow Events 382
Property Page Events 391
Publishing Events 394
Vault Events 399
Workflow Definition Events 403
executable files
Acad.exe 65
ConfiguratorU.exe 65
Explorer.exe 65
PowerUserU.exe 65
w3wp.exe 64
W3WP.exe 65
extensions 62

F

files
DLL files See *DLL files*
executable files See *executable files*
.dat files See *.dat files*
Folder Generic Events 376
formatting
text 418
functions
about
creating custom functions 292
Introduction to Functions 250
RTF codes 418
functions not supported in PowerWeb 416
types
AIMS_Commands function 251
AIMS_Properties function 252
AIMS_UpdateChangeManagement function 253
AMCreateObject function 254
AMMGetCustomColumnHeaders function 255
AMMGetCustomColumnValues function 256
AMMGetReportCustomHeaderValues function 261
AMMGetReportTableRowValues function 262
AMMMainTagDocumentId function 267
AMMPropertiesToBeRequested function 269
AMMTags4TagPageIsVisible function 270
AMMTagsManageLinksIsAllowed function 271
AMMTagsPageIsVisible function 272
AMMUseMainTag function 273
AMMWhereUsedManageLinksIsAllowed function 274
AMMWhereUsedPageIsVisible

function 275
CreateObject function 413
DebugAssert function 276
FileExtension function 277
FileRoot function 278
FormatSequenceAlfa function 279
FormatSequenceAlfaNum function 280
FormatSequenceNum function 281
GMTTime2Local function 282
ListFromColumn function 283
LocalTime2GMT function 284
PnIDLink_GetAssetCoordinates function 285
PnIDLink_GetSheetSize function 286
PnIDLink_IsMainDrawing function 287
Quote function 288
ValidateFolderName function 289
WinInputBox function 290
WinMsgBox function 291

G

global variables 415

H

hotspots
TagExtractor Object 431

L

limiting
generated events 299
log files
view in PowerWeb 66

M

Meridian Enterprise
[Meridian.ShortName] Object Model See *objects*
Meridian Enterprise Script Editor 57
messages
automatically generated 411

O

object arguments 415
Object Browser 60**,** 68**,** 407
Object Model See *objects*
objects
about 68
arguments 415
Attachment Object 69
Attachments object 70
automation objects 413
Batch Object 72
BCPropStorage Object 83
Client Object 86
Document Object 94
Methods 107
Properties 95
DocumentType Object 137
ExportPackage Object 139
ExportPackages Object 147

Folder Object 149
Methods 156
Properties 150
ImportPackage Object 165
ImportProfile Object 167
MailMessage Object 169
MailRecipient Object 171
MailRecipients Object 173
MasterDocument Object 176
MeridianQueue Object 177
MeridianTask Object 182
NewMailMessage Object 185
ProjectCopy Object 187
Query Object 189
References Object 191
Report Object 195
Roles Object 197
Scope Object 200
Sequence Object 202
StaticCollection Object 204
Table Object 206
Tag Object 438
TagCollection Object 437
TagExtractor Object 431
TagIterator Object 440
Task Object 215
User Object 217
Vault Object 222
Viewer Object 241
WaitingList Object 242
Workflow Object 244
WorkflowState Object 246
WorkflowTransition Object 248
order of events 296

P

Package Events 379
PowerWeb
VBScript functions 416
view log 66
web server
debug 65
Project Workflow Events 382
properties
specific names of
UserName 184
property events
effects of custom scripting 62
Property Page Events 391
property values
clearing property values of copied documents 410
setting based on workflow transitions 411
validating unique 410
publishing
options
VBScript
CDWF 446
DATAONLY 446
DELETE 446
LIB 446
NOREDLINES 447
NORELEASE 447

OVERWRITE 447
PUBLISHONLY 448
RENDERLAYOUTS 449
RESERVE 450
RETIRE 450
RETRY 450
SITE 450
UNRESERVE 450
Publishing Events 394
publishing jobs
options 446
PublishingCapability.dat
VBScript
publishing and rendering options 446

R

rendering options
VBScript
NOPOSTRENDER 451
NORENDER 451
_PRINTING_QUALITY_ 451
REVITSETUP 451
ExternalUpdateProperties 451
RTF codes 418

S

Script Editor See *VBScript, Meridian Enterprise Script Editor*
scripts
Object Browser 60
security
effects of custom scripting 62
requirements
debugging VB Script 63
user administration See also *user accounts*
settings
property values 411

T

TagExtractor
Result Codes 445
Tag Object 438
TagCollection Object 437
TagExtractor Component 430
TagExtractor Object 431
TagIterator Object 440
technical support
about 56
text formatting 418
title blocks
and VBScript
CAD link events 311
debugging title block links 65
publishing and rendering options 449
links
debug 65
property page 391

U

user accounts See also *security, user administration*

using the GetUsers method 238

user groups See also *security, user administration*

UserName property 184

V

vault

settings See *vault settings*

Vault Events 399

vault settings

AMM Settings

AMMMainTagDocumentId function 267

VBScript

apply True or False to a field 67

Asset Management

Tag Object 438

TagCollection Object 437

TagExtractor 430

TagExtractor Object 431

TagExtractor Result Codes 445

TagIterator Object 440

calculate a value 67

command identifiers 428

configuration expressions 67

debugging 63

effects of custom scripting 62

enable logging 65

examples

access properties and methods of an object 413

clearing property values of copied documents 410

create a connection to an Access database 414

create a TagExtractor Object 431

create automation object 413

formatting with RTF codes 419

get Excel version number on remote network computer 413

obtaining email information 415

sending email messages 411

setting property values based on workflow transitions 411

validating unique property values 410

generated events 299

Meridian Enterprise Script Editor 57

Object Browser 60

PowerWeb 416

publishing and rendering options 446

requirements 63

VBScript Events See *events*

VBScript Functions See *functions*

VBScript Objects See *objects*

Visual Studio 63

W

Web Access See *PowerWeb*

WinInputBox function

RTF codes 418

WinMsgBox function
RTF codes 418
Workflow Definition Events 403

MERIDIAN 2026 ENTERPRISE VBSCRIPT API REFERENCE – SEPTEMBER 2026

Accruent, LLC
11500 Alterra Parkway
Suite 110
Austin, TX 78758
www.accruent.com